A Robust Neural Network Approach to Optimal Decumulation and Factor Investing in Defined Contribution Pension Plans

by

Marc A. Chen

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Data Science

Waterloo, Ontario, Canada, 2023

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Marc Chen was the sole author of Chapters 2 and 5, which were written under the supervision of Drs. Peter Forsyth and Yuying Li, and were not written for publication.

Chapters 3 and 4 consist of material from a manuscript that was written for publication. This manuscript was co-first authored by Marc Chen and Mohammad Shirazi and co-authored by Drs. Peter Forsyth and Yuying Li. The research was conducted at the University of Waterloo under the supervision of Drs. Peter Forsyth and Yuying Li. Marc Chen and Mohammad Shirazi were the primary coders for the framework presented in the manuscript. The manuscript was primarily drafted by Marc Chen and Mohammad Shirazi. Drs. Peter Forsyth and Yuying Li provided intellectual input, provided feedback, and edited final versions of the manuscript.

Citation:

M. Chen, M. Shirazi, P. Forsyth, and Y. Li. "Machine Learning and Hamilton-Jacobi-Bellman Equation for Optimal Decumulation: a Comparison Study." (2023). *ArXiv* preprint. URL https://arxiv.org/abs/2306.10582.

Abstract

In this thesis, we propose a novel data-driven neural network (NN) optimization framework for solving an optimal stochastic control problem under stochastic constraints. The NN utilizes customized output layer activation functions, which permits training via standard unconstrained optimization. The optimal solution of the two-asset problem yields a multi-period asset allocation and decumulation strategy for a holder of a defined contribution (DC) pension plan. The objective function of the optimal control problem is based on expected wealth withdrawn (EW) and expected shortfall (ES) that directly targets left-tail risk. The stochastic bound constraints enforce a guaranteed minimum withdrawal each year. We demonstrate that the data-driven NN approach is capable of learning a near-optimal solution by benchmarking it against the numerical results from a Hamilton-Jacobi-Bellman (HJB) Partial Differential Equation (PDE) computational framework. The NN framework has the advantage of being able to scale to high dimensional multi-asset problems, which we take advantage of in this work to investigate the effectiveness of various factor investing strategies in improving investment outcomes for the investor.

Acknowledgements

Thank you to my family and my friends for supporting me and taking joy in my journey. Thank you to my advisers Yuying Li and Peter Forsyth for guiding me and challenging me to discover my own capability.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Access to traditional defined benefit (DB) pension plans continues to disappear for employees. In 2022, only 15% of private sector workers in the United States had access to a defined benefit plan, while 66% had access to a defined contribution (DC) plan (U.S. Bureau of Labor Statistics, 2022). In other countries, DB plans have become a thing of the past.

Defined contribution plans leave the burden of creating an investment and withdrawal strategy to the individual investor, which Nobel Laureate William Sharpe referred to as "the nastiest, hardest problem in finance" (Ritholz, 2017). Indeed, a review of the literature on decumulation strategies (Bernhardt and Donnelly, 2018; MacDonald et al., 2013) shows that balancing all of retirees' concerns with a single strategy is exceedingly difficult. As DC plans' ubiquity has increased, so has economic uncertainty: all measures of business uncertainty surveyed by the Federal Reserve remain significantly elevated compared to the years preceding the Covid-19 pandemic (Federal Reserve Bank of Atlanta, 2023). Any tools investors have to mitigate risk in DC plans are of great interest. This thesis responds to this need by 1) developing a data-driven neural network framework that closely approximates the numerical solution to the optimal decumulation problem for DC plans and 2) extending the framework to incorporate the increasingly popular paradigm of factor investing, which has been used by asset managers to diversify portfolios away from the broad market.

Our starting point is to build on the approach in Forsyth (2022). This work seeks to address retirees' goal of finding an optimal balance between maximizing withdrawals and minimizing the risk of depletion while guaranteeing a minimum withdrawal. The problem is formulated as one in optimal stochastic control and determines a decumulation and allocation strategy for a standard 30-year investment horizon. Numerical solutions

are obtained using dynamic programming, which results in a Hamilton-Jacobi-Bellman (HJB) Partial Differential Equation (PDE). The HJB PDE framework developed in Forsyth (2022) maximizes expected withdrawals and minimizes the risk of running out of savings, measured by the left tail in the terminal wealth distribution. Since maximizing withdrawals and minimizing risk are conflicting measures, we use a scalarization technique to compute Pareto optimal points. A constant lower bound is imposed on the withdrawal, providing a guaranteed income. An upper bound on withdrawal is also imposed, which can be viewed as the target withdrawal. The constraints of no shorting and no leverage are imposed on the investment allocation.

The solution to this constrained stochastic optimal control problem yields a dynamic stochastic strategy, naturally aligning with retirees' concerns and objectives. Note that cash flows are not mortality weighted, consistent with Bengen (1994). This can be justified on the basis of *planning to live, not planning to die* as discussed in Pfau (2018).

Our dynamic strategy can be contrasted to traditional strategies such as the *Bengen Rule* (4% Rule), which recommends withdrawing a constant 4% of initial capital each year (adjusted for inflation) and investing equal amounts into stocks and bonds (Bengen, 1994). Initially proposed in 1994, Scott et al. (2009) found the 4% Rule to still be a popular strategy 14 years later, and the near-universal recommendation of the top brokerage and retirement planning groups. Recently there has been acknowledgment in the asset management industry that the 4% Rule is sub-optimal, but wealth managers still recommend variations of the same constant withdrawal principle (Williams and Kawashima, 2023). The strategy proposed by Forsyth (2022) is shown to be far more efficient than the Bengen 4% Rule. Of course, the PDE solution in Forsyth (2022) is restricted to low dimensions (i.e. a small number of stochastic factors). In order to remedy some of the deficiencies of PDE methods (such as in Forsyth (2022)), we propose a neural network (NN) based framework without using dynamic programming. The NN framework we develop here builds upon the work of Li and Forsyth (2019) by creating a formulation that simultaneously solves a control for the withdrawal in addition to the allocation.

In contrast to the PDE solution approach, our proposed NN approach has the following advantages:

(i) It is data-driven and does not depend on a parametric model. This makes the framework versatile in selecting training data, and less susceptible to model misspecification.

(ii) The control is learned directly, thereby exploiting the low dimensionality of the control (Van Staden et al., 2023). This technique thus avoids dynamic programming and

the associated error propagation. The NN approach can also be applied to higher dimensional problems, such as those with a large number of assets.

(iii) If the control is a continuous function of time, the control approximated by the NN framework will reflect this property. If the control is discontinuous [1], the NN seems to produce a smooth, but quite accurate, approximation.[2]

Since the NN only generates an approximate solution to the complex stochastic optimal control problem, it is essential to assess its accuracy and robustness. Rarely is the quality of an NN solution assessed rigorously, since an accurate solution to the optimal control problem is often not available. In Chapter 4, we compare the NN solution to the decumulation problem against the ground-truth results from the provably convergent HJB PDE method.

While other neural network and deep learning methods for optimal stochastic control problems have been proposed before, they differ significantly from our approach in their architecture, taking a *stacked* neural network approach as in Buehler et al. (2019); Chen and Wan (2021); Han and E (2016); Tsang and Wong (2020) or a hybrid dynamic programming and reinforcement learning approach (Huré et al., 2021). On the other hand, our framework uses the same two neural networks at all rebalancing times in the investment scenario. Since our NNs take time as an input, the solution will be continuous in time if the control is continuous. Note that the idea of using time as an input to the NN was also suggested in Laurière et al. (2021). According to the taxonomy of sequential decision problems proposed in Powell (2021), our approach would most closely be described as Policy Function Approximation (PFA).

With the exception of Laurière et al. (2021), previous works do not provide a benchmark for numerical methods, as we do in this work. Our results show that our proposed NN method is able to approximate the numerical results in Forsyth (2022) with high accuracy. Especially notable, and somewhat unexpected, is that the *bang-bang* control[3] for the withdrawal is closely reproduced by the NN method.

As machine learning and artificial intelligence based methods continue to proliferate in finance and investment management, it is crucial to demonstrate that these methods are reliable and explainable (Boukherouaa et al., 2021). We believe that our proposed framework and test results make a step forward in demonstrating deep learning's potential for stochastic control problems in finance.

[1] *Bang-bang* controls, frequently encountered in optimal control, are discontinuous as a function of time.
[2] For a possible explanation of this, see Ismailov (2023).
[3] In optimal stochastic control, a bang-bang control is a discontinuous function of the state.

After establishing the reliability of the NN framework, we extend it to an investigation of the utility of factor investing for the decumulation problem. Factor investing refers to the investment approach that aims to expose a portfolio to quantifiable firm characteristics ("factors"), such as market capitalization or book-to-market ratio. This is done to "tilt" the portfolio away from the broad market in hopes of mitigating risk or finding excess returns. As the factor investing paradigm has quickly become more popular in the asset management industry, we utilize the opportunity to employ the novel NN framework developed in this thesis to investigate its potential efficacy in improving withdrawal efficiency and risk management in the decumulation problem.

To summarize, the main contributions of this thesis are as follows:

- Proposing an NN framework with suitable activation functions for decumulation and allocation controls, which yields an approximate solution to the constrained stochastic optimal decumulation problem in Forsyth (2022) by solving a standard unconstrained optimization problem;
- Demonstrating that the NN solution achieves very high accuracy in terms of the efficient frontier and the decumulation control when compared to the solution from the provably convergent HJB PDE method. The NN framework is also shown to generate strategies with robust performance on out-of-sample and out-of-distribution data, even when trained on alternate data sets;
- Illustrating that, with a suitably small regularization parameter, the NN allocation strategy can differ significantly from the PDE allocation strategy in the region of high wealth and near the terminal time, while the relevant performance statistics remain unaffected. This is due to the fact that the problem is ill-posed in these regions of the state space unless we add a small regularization term;
- Extending the novel NN framework to the dynamic factor investing problem. We find promising results for the potential of NN strategies that incorporate factor-based assets in providing improved performance compared to portfolios with only basic assets or constant allocation strategies.

1.1 Thesis Outline

Chapter 2 briefly reviews the theoretical background of optimal control, neural network approximation, and factor theory; Chapter 3 develops the basic asset decumulation problem

formulation and NN framework; Chapter 4 discusses model results and establishes the accuracy of the NN method compared to the HJB method; finally, Chapter 5 extends the framework to include dynamic factor investing.

Chapter 2

Background

2.1 Optimal Control Theory

Given a *system*, optimal control is the process of finding the control policy that maximizes a specific performance criterion. We provide a brief overview of the general problem formulation for optimal control in this section. For a more rigorous treatment, refer to books such as Sethi (2019) or Kirk (2016).

A state variable $X(t)$ can be chosen to represent the system at any time $t \in [0,T]$, where $T > 0$ is the specified time horizon for the system we are considering. For example, in the investment problems dealt with in this work, $X(t)$ will include amounts invested in a portfolio's assets at time t.

In a meaningful optimal control problem, there will be aspects of the system that we can and cannot influence. The parts of the system that we can control are the *control variables*, which we will collectively refer to as $\mathcal{P} = \{p(t), t \in [0,T]\}$, where $p(t)$ is the control for a specific time t. In our case, we will be able to formulate control variables that dictate how to reallocate money in a portfolio, but we will always be subject to the processes that evolve $X(t)$ that we cannot control, such as the movement of asset prices. In optimal control, our goal is to select such a control policy $\mathcal{P}$, subject to any set of constraints $\mathcal{Z}$, that interacts with the system over the time horizon to maximize a predetermined objective function, J. In general, the objective function will have a component that is only dependent on the terminal state of the system as well as a component that is path-dependent. When the system is deterministic, this general objective can be stated as such (Sethi, 2019):

$$J = \int_0^T F(X(t), p(t), t)dt + S(X(T), T), \tag{2.1.1}$$

where F is a function of x, t, and $p(t)$, In order to optimize the objective, optimal control problems deal with determining the equation that characterizes the system state's evolution:

$$\frac{dx}{dt}(t) = f(X(t), p(t), t), \;\; X(0) = x_0 \tag{2.1.2}$$

Solving optimal control problems can be viewed as an extension to calculus since we are interested in what value of $\mathcal{P}$ will influence $X(t)$ so that J is maximized. In summary, problems of this nature can be expressed in the form:

$$\begin{cases} \max\limits_{\mathcal{P}\in\mathcal{Z}} \left[J = \int_0^T F(X(t), \mathcal{P}, t)dt + S(X(T), T) \right] \\ \text{subject to} \\ \frac{dx}{dt}(t) = f(x(t), p(t), t), \;\; x(0) = x_0. \end{cases} \tag{2.1.3}$$

2.2 Stochastic Optimal Control

Many optimal control problems consider systems that are deterministic, where the equation governing the evolution of the system can be known with certainty. In the problems considered in this work, this will not be the case. We will consider systems whose evolution is influenced by stochastic processes and we will therefore need to describe and analyze the system with stochastic mathematics. In the following, we briefly review the general problem formulation for stochastic optimal control and the intuition behind the resulting Hamilton-Jacobi-Bellman (HJB) equation that can be used in a dynamic programming (DP) framework to solve such optimization problems. We will assume continuous time and standard Brownian motion to model stochasticity, however, in the later chapters of this thesis, we will deal with the discrete time case, which is described in Bertsekas and Shreve (1996) and consider problems that include stochastic processes with discrete jumps, which are described in Fleming and Rishel (1975) and Davis and Farid (1999).

In *stochastic optimal control*, we seek a control that optimizes an objective constructed as the *expectation* of the chosen performance criterion (Bertsekas and Shreve, 1996):

$$J = E\left[\int_0^T F(X(t), p(t), t)dt + S(X(T), T)\right] \tag{2.2.1}$$

Since the state equation is now governed by random processes, we write it as:

$$dX(t) = f(X(t), p(t), t)dt + G(X(t), p(t), t)dZ(t), \;\; X(0) = x_0, \tag{2.2.2}$$

where $Z(t)$ represents a standard Wiener process. Therefore, f represents the deterministic drift of the state equation, while G represents the (stochastic) diffusion. For the purposes of describing the background of this thesis, we assume that both of these function components and objective components F and S are continuously differentiable. To solve the problem defined by (2.2.1) and (2.2.2), we first define the *value function* (sometimes also referred to as the "cost-to-go" function), $V(x,t)$, to be the expected value of the objective when the optimal policy is followed from time t to T. By defining it recursively, we express $V(x,t)$ at any time t:

$$V(x,t) = \max_{p(t)} E\big[F(X(t), p(t), t)dt + V(x(t) + dX(t), t + dt)\big] \tag{2.2.3}$$

We can approximate V using its Taylor expansion:

$$\begin{aligned} V(x + dX, t + dt) = V(x,t) + V_t dt + V_x dX + \frac{1}{2}V_{xx}(dX)^2 + V_{tt}(dt)^2 \\ + \frac{1}{2}V_{xt}dX_t dt + ... \end{aligned} \tag{2.2.4}$$

The dX terms in (2.2.4) can be written in terms of f and G by starting from (2.2.2):

$$\begin{aligned} (dX)^2 = f^2(dt)^2 + G^2(dZ)^2 + 2fGdZdt, \\ dXdt = f(dt)^2 + GdZdt. \end{aligned} \tag{2.2.5}$$

The following expressions come from the theory of stochastic calculus, and allow us to differentiate geometric Brownian motion; see for example Karatzas and Shreve (1991).

$$(dZ)^2 = dt,\ dZdt = 0,\ dt^2 = 0. \tag{2.2.6}$$

Substituting (2.2.6) and (2.2.5) into (2.2.4) and using the fact that $E[dZ] = 0$ yields:

$$V(x + dX, t + dt) = V(x,t) + V_t dt + V_x f dt + \frac{1}{2} V_{xx} G^2 dt, \tag{2.2.7}$$

which allows us to substitute the recursive term in (2.2.8). After canceling the $V(x,t)$ terms and dividing by dt, we get:

$$0 = \max_{p(t)} E\big[F + V_t + V_x f + \frac{1}{2} V_{xx} G^2\big], \tag{2.2.8}$$

which is the HJB equation for the value function $V(x,t)$ with terminal condition $V(x,T) = S(x,T)$.

In computational practice, the HJB equation is solved backward in time, starting from the terminal condition and solving for the optimal $p(t)$ at each time step t until the beginning of the period is reached. This process is dynamic programming; breaking up the larger problem into many smaller problems until the entire problem can be solved. Forsyth (2022) uses an HJB method to develop a numerical algorithm for the decumulation problem, formulated as a problem in stochastic optimal control.

There has been significant research on replacing the value function with a sufficiently accurate approximation at each time step. In the next section, we will briefly introduce neural network approximation theory and its applications in optimal control.

2.3 Neural Networks and Function Approximation

Beginning in the 1980s, there has been significant research into how to reduce the computational complexity of stochastic optimal control problems by using sufficiently accurate approximations of the value function. Bertsekas and Tsitsiklis (1996) first dubbed the use of neural networks for this class of problems as 'neuro-dynamic programming' and Sutton (1988) established this as the beginning of modern computational reinforcement learning (RL). In this section, we briefly introduce the theory of neural network (NN) approximation and its application to optimal control. For a more complete discussion on NNs and

deep learning refer to books such as Goodfellow et al. (2016). The NN frameworks developed in this thesis and Li and Forsyth (2019) (upon which this thesis builds), differ from 'neuro-dynamic programming' and RL approaches by using NN approximation in a fundamentally different way, approximating the control directly instead of the value function and eliminating the need for anything resembling dynamic programming.

2.3.1 Basic NN Formulation

Neural networks (NNs) are machine learning models that consist of layers of interconnected 'nodes.' Each node receives values from all incoming connections, processes the values, and then sends its result to all of its outgoing connections. 'Learning' occurs by adjusting the weights that are applied to each connection, as well as the biases that are added as part of each node's input processing. There are many learning algorithms that can be employed for this. A basic NN's architecture must be adapted to the problem at hand; including changing the number of interconnected layers, the number of nodes in each layer, *activation functions* used to process nodes' input values, and so on.

Consider a NN with $\mathcal{L}$ hidden layers, which is fully connected and feed-forward. The layers are indexed by $\ell \in \{0, \ldots, \mathcal{L}+1\}$, with $\ell = 0$ being the input layer and $\ell = \mathcal{L}+1$ being the output layer. Let $\eta_\ell \in \mathbb{N}$ be the number of nodes in layer ℓ. The number of nodes in the input layer, η_0, is the same as the number of elements in the feature vector, $\phi \in \mathbb{R}^0$. Each node in the input layer takes a value from the feature vector.

The connections going into each layer except the input layer have a matrix of weights associated with them, $x^{[\ell]} \in \mathbb{R}^{\eta_{\ell-1} \times \eta_\ell}$. Each non-input layer also has a vector of biases, $b^{[\ell]} \in \eta_\ell$ and activation function, $\sigma_{[\ell]} : \mathbb{R}^{\eta_\ell} \hookrightarrow \mathbb{R}^{\eta_\ell}$, which is applied to the sum of the layer's weighted inputs, $z^{[\ell]}$. The output of the ℓ-th layer and is thus:

$$a^\ell = \sigma_{[\ell]}\big(z^{[\ell]}\big), \text{ where } z_i^{[\ell]} = \left(\sum_{k=1}^{\eta_{\ell-1}} x_{ki}^{[\ell]} a_k^{[\ell-1]}\right) + b_i^{[\ell]}, \; l \in \{1, \ldots, \mathcal{L}+1\}. \tag{2.3.1}$$

The choice of $\sigma_{[\ell]}$ is most commonly a logistic sigmoid function or the softmax function, although in this thesis we will also use novel customized versions of these functions to adapt the NN to the requirements of our problem. We will leave the discussion of how this general NN formulation is adapted to our control problem in later chapters.

2.3.2 Universal Approximation Theorems

A remarkable quality of the simple NN formulation discussed above is that neural networks based on this can be shown to be universal approximators. The earliest of such Universal Approximation theorems are Cybenko (1989) and Hornik et al. (1989), which show that NNs of arbitrary width and a single hidden layer can represent approximations of any continuous function to arbitrary accuracy, given appropriate weight and bias parameters. It should be noted that these theorems do not guarantee that a learning algorithm will always be able to learn the correct weights and biases.

It will also become relevant over the course of this thesis that research into the ability of NNs to accurately represent or learn discontinuous functions is an area of active research (Della Santa and Pieraccini, 2023; Ismailov, 2023). In this work, we find that our proposed NN framework is able to learn a discontinuous "bang-bang" control policy, which is a discontinuous function of feature inputs.

2.4 Factor Theory

The primary aim of factor theory is to create models of quantifiable characteristics that explain asset returns. The oldest and most foundational model of stock returns is the Capital Asset Pricing Model (CAPM), proposed by Sharpe (1964), where the market itself is the only systematic factor driving returns and the residual is explained by the specific asset itself:

$$r_i = r_f + \beta_i(r_M - r_f) + \varepsilon_i \tag{2.4.1}$$

where:
r_i = return of asset i
r_f = risk-free rate
β_i = beta for asset i
r_M = market factor return
ε_i = return specific to asset i, modeled as $N(0,\sigma_i^2)$, where specific risk of asset i is σ_i.

In CAPM, there is one assumed factor: the market itself, which is modeled independently from the specific asset return, ε_i. The 'beta' that gives how sensitive asset i is to the market factor is given by:

$$\beta_i = \frac{\rho_{i,M}\sigma_i}{\sigma_M}. \tag{2.4.2}$$

where:
$\rho_{i,M}$ = the correlation between asset i and the market
σ_i = volatility of asset i
σ_M = volatility of market.

In CAPM risk analysis, the beta of each asset in a portfolio is calculated to determine the beta of the entire portfolio. When a portfolio is sufficiently diversified, the specific risk of assets is effectively eliminated and only the systematic risk remains, which consists of the portfolio beta and market beta. Taking the expectation, specific risk can be zeroed out since $E[\varepsilon_i] = 0$. The CAPM model can then be rewritten to provide an expectation of excess returns:

$$E[r_i - r_f] = \beta_i(E[r_M] - r_f). \tag{2.4.3}$$

In (2.4.3), the systematic return of asset i is modeled only in terms of the market factor. While useful and foundational in asset pricing, strong evidence of pricing anomalies was discovered that contradicted this model. The Fama-French 3 Factor Equity Model introduced firm size and book-to-market ratio as factors in an asset pricing model, which was widely regarded as an improvement over CAPM. The work of Nobel Prize winner Eugene Fama and Kenneth French (Fama and French, 1993) was also a seminal paper in the field of factor research and spawned work on many subsequent models of asset returns including a vast array of proposed factors. These expanded factor models have the form of

$$E[r_i - r_f] = \beta_{i,1}E\big[E[r_1] - r_f\big] + \beta_{i,2}E\big[E[r_2] - r_f\big] + \dots \tag{2.4.4}$$

where:
r_k = the return of factor k
$\beta_{i,k}$ = the beta of asset i with respect to factor k.

Factor research seeks to identify factors that drive asset performance and explain the cross-section of returns by using models of the form (2.4.4), or variations of it, for empirical

experiments using historical asset returns. A huge amount of this kind of research has been done, resulting in what some call the "zoo" of factors (Feng et al., 2020). Factor researchers face a daunting task: the process of identifying candidate factors and testing is complex and much of the existing research is biased towards positive results, often muddying the waters. The Fama-French 3-Factor Model has been updated by newer models with new factors that have been widely accepted, however, such as the 4-Factor Model proposed by Carhart (1997), which introduced the Momentum factor. We leave the definition of specific factors relevant to this thesis to Chapter 5.

Given the complexity of factor research and its rise in popularity coinciding with financial applications of machine learning becoming more accepted, there have naturally been many researchers utilizing machine learning models in this work. For example, Feng et al. (2022) provides a deep learning framework to generate factors based on an economic-guided objective function formulated to minimize pricing errors. The book (Coqueret and Guida, 2023) provides an overview of the methods used in machine learning to identify factors. The application of machine learning in this thesis differs significantly from this kind of research. This thesis instead relies on widely accepted and established factors as proxies for factor-based ETF assets that a retiree may choose to invest in as a means to mitigate market risk (diversifying away from market beta) and seek returns exceeding the market. The NN-based approach we propose uses these factor assets as part of an NN-generated strategy to optimally decumulate and allocate a portfolio.

Chapter 3

Portfolio Decumulation

In this chapter, we begin by describing the mathematical formulation of optimal portfolio decumulation as a problem in optimal stochastic control. We then describe two distinct optimization frameworks to solve the same problem: 1) A dynamic programming approach based upon the Hamilton-Jacobi-Bellman equation and 2) a novel data-driven neural network approach. The results for these two methods are then analyzed and compared in the following chapter.

3.1 Problem Formulation

3.1.1 Overview

The investment scenario described in Forsyth (2022) concerns an investor with a portfolio wealth of a specified size, upon retirement. The investment horizon is fixed with a finite number of equally spaced rebalancing times (usually annually). At each rebalancing time, the investor first chooses how much to withdraw from the portfolio and then how to allocate the remaining wealth. The investor must withdraw an amount within a specified range. The wealth in this portfolio can be allocated to any mix of two given assets, with no shorting or leverage. The assets the investor can access are a broad stock index fund and a constant maturity bond index fund.

In the time that elapses between re-balancing times, the portfolio wealth will change according to the dynamics of the underlying assets. If the wealth of the portfolio goes below zero, the portfolio is liquidated, trading ceases, debt accumulates at the borrowing

rate, and withdrawals are restricted to the minimum amount. At the end of the time horizon, a final withdrawal is made and the portfolio is liquidated, yielding the terminal wealth.

We assume here that the investor has other assets, such as real estate, which are non-fungible with investment assets. These other assets can be regarded as a hedge of last resort, which can be used to fund any accumulated debt (Pfeiffer et al., 2013). This is not a novel assumption and is in line with the mental bucketing idea proposed by Shefrin and Thaler (1988). The use of this assumption within literature targeting similar problems is also common (see Forsyth et al. (2022)). Of course, the objective of the optimal control is to make running out of savings an unlikely event.

The investor's goal then is to maximize the weighted sum of total withdrawals and the mean of the worst 5% of the outcomes (in terms of terminal wealth). We term this tail risk measure as Expected Shortfall (ES) at the 5% level. In this section, this optimization problem will be described with the mathematical details common to both the HJB and NN methods.

3.1.2 Stochastic Process Model

Let S_t and B_t represent the real (i.e. inflation-adjusted) *amounts* invested in the stock index and a constant maturity bond index, respectively. These assets are modeled with correlated jump diffusion models, in line with MacMinn et al. (2014). These parametric stochastic differential equations (SDEs) allow us to model non-normal asset returns. The SDEs are used in solving the HJB PDE, and generating training data with Monte Carlo simulations in the proposed NN framework. For the remainder of this thesis, we refer to simulated data using these models as synthetic data.

When a jump occurs, $S_t = \xi^s S_{t^-}$, where ξ^s is a random number representing the jump multiplier and $S_{t^-} = S(t-\epsilon), \epsilon \to 0^+$ (S_{t^-} is the instant of time before t). We assume that $\log(\xi^s)$ follows a double exponential distribution (Kou, 2002; Kou and Wang, 2004). The jump is either upward or downward, with probabilities u^s and $1-u^s$ respectively. The density function for $y = \log(\xi^s)$ is

$$f^s(y) = u^s \eta_1^s e^{-\eta_1^s y} \mathbf{1}_{y \geq 0} + (1-u^s)\eta_2^s e^{\eta_2^s y} \mathbf{1}_{y<0} \ . \tag{3.1.1}$$

We also define

$$\gamma_\xi^s \quad = \quad E[\xi^s - 1] = \frac{u^s \eta_1^s}{\eta_1^s - 1} + \frac{(1-u^s)\eta_2^s}{\eta_2^s + 1} - 1 \ . \tag{3.1.2}$$

The starting point for building the jump diffusion model is a standard geometric Brownian motion, with drift rate μ^s and volatility σ^s. A third term is added to represent the effect of jumps, and a compensator is added to the drift term to preserve the expected drift rate. For stocks, this gives the following stochastic differential equation (SDE) that describes how S_t (inflation adjusted) evolves in the absence of a control:

$$\frac{dS_t}{S_{t^-}} = \left(\mu^s - \lambda_\xi^s \gamma_\xi^s\right) dt + \sigma^s \, dZ^s + d\left(\sum_{i=1}^{\pi_t^s} (\xi_i^s - 1)\right), \tag{3.1.3}$$

where dZ^s is the increment of a Wiener process, π_t^s is a Poisson process with positive intensity parameter λ_ξ^s, and $\xi_i^s \ \forall i$ are i.i.d. positive random variables having distribution (3.1.1). Moreover, ξ_i^s, π_t^s, and Z^s are assumed to all be mutually independent.

As is common in the practitioner literature, we directly model the returns of a constant maturity (inflation adjusted) bond index by a jump diffusion process (Lin et al., 2015; MacMinn et al., 2014). Let the amount in the constant maturity bond index be $B_{t^-} = B(t-\epsilon), \epsilon \to 0^+$. In the absence of a control, B_t evolves as

$$\frac{dB_t}{B_{t^-}} = \left(\mu^b - \lambda_\xi^b \gamma_\xi^b + \mu_c^b \mathbf{1}_{\{B_{t^-}<0\}}\right) dt + \sigma^b \, dZ^b + d\left(\sum_{i=1}^{\pi_t^b} (\xi_i^b - 1)\right), \tag{3.1.4}$$

where the terms in Equation (3.1.4) are defined analogously to Equation (3.1.3). In particular, π_t^b is a Poisson process with positive intensity parameter λ_ξ^b, $\gamma_\xi^b = E[\xi^b - 1]$, and $y = \log(\xi^b)$ has the same distribution as in equation (3.1.1) (denoted by $f^b(y)$) with distinct parameters, u^b, η_1^b, and η_2^b. Note that ξ_i^b, π_t^b, and Z^b are assumed to all be mutually independent, as in the stock SDE. The term $\mu_c^b \mathbf{1}_{\{B_{t^-}<0\}}$ represents the extra cost of borrowing (a spread).

The correlation between the two assets' diffusion processes is ρ_{sb}, giving us $dZ^s \cdot dZ^b = \rho_{sb} dt$. The jump processes are assumed to be independent. For further details on the justification of this market model, refer to Forsyth (2022).

We define the investor's total wealth at time t as

$$\text{Total wealth} \equiv W_t = S_t + B_t. \tag{3.1.5}$$

Barring insolvency, shorting stock and using leverage (i.e., borrowing) are not permitted, a realistic constraint in the context of DC retirement plans. Furthermore, if the wealth

ever goes below zero, the portfolio is liquidated, trading ceases, and debt accumulates at the borrowing rate. We emphasize that we are assuming that the retiree has other assets (i.e., residential real estate) which can be used to fund any accumulated debt. In practice, this could be done using a reverse mortgage (Pfeiffer et al., 2013).

3.1.3 Notational Conventions

We define the finite set of discrete withdrawal/rebalancing times $\mathcal{T}$,

$$\mathcal{T} = \{t_n = n\Delta t | n = 0, \ldots, M\}, \quad \Delta t = T/M \ . \tag{3.1.6}$$

The beginning of the investment period is $t_0 = 0$. We assume each rebalancing time is evenly spaced, meaning $t_n - t_{n-1} = \Delta t = T/M$ is constant. To avoid subscript clutter in the following, we will occasionally use the notation $S_t \equiv S(t), B_t \equiv B(t)$ and $W_t \equiv W(t)$. At each rebalancing time, $t_n \in \mathcal{T}$, the investor first withdraws an amount of cash q_n from the portfolio and then rebalances the portfolio. At time T, there is one final withdrawal, q_T, and then the portfolio is liquidated. We assume no taxes, which is reasonable since retirement accounts are typically tax-advantaged. In addition, since trading is infrequent, we assume transaction costs to be negligible (Dang and Forsyth, 2014). Given an arbitrary time-dependent function, $f(t)$, we will use the shorthand

$$f(t_n^+) \equiv \lim_{\epsilon \to 0^+} f(t_n + \epsilon) \ , \qquad f(t_n^-) \equiv \lim_{\epsilon \to 0^+} f(t_n - \epsilon) \ . \tag{3.1.7}$$

The multidimensional controlled underlying process is denoted by $X(t) = (S(t), B(t))$, $t \in [0,T]$. For the realized state of the system, $x = (s, b)$.

At the beginning of each rebalancing time t_n, the investor withdraws the amount $q_n(\cdot)$, determined by the control at time t_n; that is, $q_n(\cdot) = q_n(X(t_n^-)) = q(X(t_n^-), t_n)$. This control is used to evolve the investment portfolio from W_t^- to W_t^+.

$$W(t_n^+) = W(t_n^-) - q_n \ , \qquad t_n \in \mathcal{T} \ . \tag{3.1.8}$$

Formally, both withdrawal and allocation controls depend on the state of the portfolio before withdrawal, $X(t_n^-)$, but it will be computationally convenient to consider the allocation control as a function of the state after withdrawal since the portfolio allocation is

rebalanced after the withdrawal has occurred. Hence, the allocation control at time t_n is $p_n(\cdot) = p_n(X(t_n^+)) = p(X(t_n^+), t_n)$.

$$p_n(X(t_n^+)) = p(X(t_n^+), t_n) = \frac{S(t_n^+)}{S(t_n^+) + B(t_n^+)} \ . \tag{3.1.9}$$

As formulated, the controls depend on wealth only (see Forsyth (2022) for a proof, assuming no transaction costs). Therefore, we make another notational adjustment for the sake of simplicity and consider $q_n(\cdot)$ to be a function of wealth before withdrawal, W_n^-, and $p_n(\cdot)$ to be a function of wealth after withdrawal, W_n^+.

We assume instantaneous rebalancing, which means there are no changes in asset prices in the interval (t_n^-, t_n^+). A control at time t_n is therefore described by a pair $(q_n(\cdot), p_n(\cdot)) \in \mathcal{Z}(W_n^-, W_n^+, t_n)$, where $\mathcal{Z}(W_n^-, W_n^+, t_n)$ represents the set of admissible control values for t_n. The constraints on the allocation control are no shorting, no leverage (assuming solvency). There are minimum and maximum values for the withdrawal. When wealth goes below zero due to withdrawals ($W_n^+ < 0$), trading ceases with debt accumulating at the borrowing rate, and withdrawals are restricted to the minimum. Stock assets are liquidated at the end of the investment period. We can mathematically state these constraints by imposing suitable bounds on the value of the controls as follows:

$$\mathcal{Z}_q(W_n^-, t_n) = \begin{cases} [q_{\min}, q_{\max}] \ ; \ t_n \in \mathcal{T} \ ; \ W_n^- > q_{\max} \\ [q_{\min}, W_n^-] \ ; \ t_n \in \mathcal{T} \ ; \ q_{\min} < W_n^- < q_{\max} \\ \{q_{\min}\} \ ; \ t_n \in \mathcal{T} \ ; \ W_n^- < q_{\min} \end{cases} , \tag{3.1.10}$$

$$\mathcal{Z}_p(W_n^+, t_n) = \begin{cases} [0,1] & W_n^+ > 0 \ ; \ t_n \in \mathcal{T} \ ; \ t_n \neq t_M \\ \{0\} & W_n^+ \leq 0 \ ; \ t_n \in \mathcal{T} \ ; \ t_n \neq t_M \\ \{0\} & t_n = t_M \end{cases} , \tag{3.1.11}$$

$$\mathcal{Z}(W_n^-, W_n^+, t_n) = \mathcal{Z}_q(W_n^-, t_n) \times \mathcal{Z}_p(W_n^+, t_n) \ . \tag{3.1.12}$$

At each rebalancing time, we seek the optimal control for all possible combinations of $(S(t), B(t))$ having the same total wealth (Forsyth, 2022). Hence, the controls for both withdrawal and allocation are formally a function of wealth and time before withdrawal (W_n^-, t_n), but for implementation purposes it will be helpful to write the allocation as a function of wealth and time after withdrawal (W_n^+, t_n). The set of admissible controls $\mathcal{A}$ can be written as

$$\mathcal{A} = \left\{ (q_n, p_n)_{0 \leq i \leq M} : (q_n, p_n) \in \mathcal{Z}(W_n^-, W_n^+, t_n) \right\} \tag{3.1.13}$$

An admissible control $\mathcal{P}$, with range in $\mathcal{A}$, can be written as

$$\mathcal{P} = \{(q_n(\cdot), p_n(\cdot)) \ : \ n = 0, \ldots, M\} \ . \tag{3.1.14}$$

It will sometimes be necessary to refer to the tail of the control sequence at $[t_n, t_{n+1}, \ldots, t_M]$, which we define as

$$\mathcal{P}_n = \{(q_n(\cdot), p_n(\cdot)) \ldots, (p_M(\cdot), q_M(\cdot))\} \ . \tag{3.1.15}$$

The essence of the problem, for both the HJB and NN methods outlined in this chapter, will be to find an optimal control $\mathcal{P}^*$.

3.1.4 Risk: Expected Shortfall

Assume that $g(W_T)$ is the probability density of terminal wealth W_T at $t = T$. Then suppose

$$\int_{-\infty}^{W_\alpha^*} g(W_T) \ dW_T = \alpha \ , \tag{3.1.16}$$

i.e., $Pr[W_T < W_\alpha^*] = \alpha$, and W_α^* is the Value at risk (VAR) at the level α. We then define the Expected Shortfall (ES) as the mean of the worst α fraction of the terminal wealth. Mathematically,

$$\text{ES}_\alpha = \frac{\int_{-\infty}^{W_\alpha^*} W_T \ g(W_T) \ dW_T}{\alpha} \ . \tag{3.1.17}$$

As formulated, a higher ES is more desirable than a smaller ES (Equation (3.1.17) is formulated in terms of final wealth not losses). To approximate ES, we use the alternate formulation as suggested by Rockafellar and Uryasev (2000),

$$\text{ES}_\alpha \ = \ \sup_{W^*} E\left[W^* + \frac{1}{\alpha}\min(W_T - W^*, 0)\right] \ . \tag{3.1.18}$$

Under a control $\mathcal{P}$, and initial state X_0, this becomes:

$$\text{ES}_\alpha(X_0^-, t_0^-) \ = \ \sup_{W^*} E_{\mathcal{P}}^{X_0^-, t_0^-}\left[W^* + \frac{1}{\alpha}\min(W_T - W^*, 0)\right] \ . \tag{3.1.19}$$

The candidate values of W^* can be taken from the set of possible values of W_T. It is important to note here that we define $\text{ES}_\alpha(X_0^-,t_0^-)$ which is the value of ES_α as seen at t_0^-. Hence, W^* is fixed throughout the investment horizon. In fact, we are considering the induced time consistent strategy, as opposed to the time inconsistent version of an expected shortfall policy (Forsyth, 2020; Strub et al., 2019). This issue is addressed in more detail in Section 3.1.7.

3.1.5 Reward: Expected Total Withdrawals

We use expected total withdrawals as a measure of reward. Mathematically, we define expected withdrawals (EW) as

$$\text{EW}(X_0^-, t_0^-) = E_{\mathcal{P}}^{X_0^-,t_0^-}\left[\sum_{n=0}^{M} q_n\right] . \tag{3.1.20}$$

Remark 3.1.1 (No discounting, no mortality weighting)**.** *Note that we do not discount the future cash flows in Equation (3.1.20). We remind the reader that all quantities are assumed real (i.e. inflation-adjusted), so that we are effectively assuming a real discount rate of zero, which is a conservative assumption. This is also consistent with the approach used in the classical work of Bengen (1994). In addition, we do not mortality weight the cash flows, which is also consistent with Bengen (1994). See Pfau (2018) for a discussion of this approach (i.e.* plan to live, not plan to die*).*

3.1.6 Defining a Common Objective Function

In this section, we describe the common objective function used by both the HJB method and the NN method.

Expected Withdrawals (EW) and Expected Shortfall (ES) are conflicting measures. We use a scalarization method to determine Pareto optimal points for this multi-objective problem. For a given κ, we seek the optimal control $\mathcal{P}_0$ such that the following is maximized,

$$\text{EW}(X_0^-, t_0^-) + \kappa \text{ES}_\alpha(X_0^-, t_0^-) . \tag{3.1.21}$$

We define (3.1.21) as the pre-commitment EW-ES problem ($PCEE_{t_0}(\kappa)$) and write the problem formally as

$(PCEE_{t_0}(\kappa))$:

$$J\left(s,b,t_0^-\right) = \sup_{\mathcal{P}_0\in\mathcal{A}}\sup_{W^*}\left\{ E_{\mathcal{P}_0}^{X_0^-,t_0^-}\left[\sum_{n=0}^{M} q_n \; + \; \kappa\left(W^* + \frac{1}{\alpha}\min(W_T - W^*, 0)\right) \overbrace{+\epsilon W_T}^{stabilization} \;\middle|\; X(t_0^-) = (s,b) \right]\right\}$$

$$\text{subject to } \begin{cases} (S_t, B_t) \text{ follow processes (3.1.3) and (3.1.4)}; \;\; t \notin \mathcal{T} \\ W_n^+ = S_n^- + B_n^- - q_n \,; \; X_n^+ = (S_n^+, B_n^+) \\ S_n^+ = p_n(\cdot)W_n^+ \,; \; B_n^+ = (1 - p_n(\cdot))W_n^+ \\ (q_n(\cdot), p_n(\cdot)) \in \mathcal{Z}(W_n^-, W_n^+, t_n) \\ i = 0, \ldots, M \; ; \; t_n \in \mathcal{T} \end{cases} \tag{3.1.22}$$

The ϵW_T stabilization term serves to avoid ill-posedness in the problem when $W_t \gg W^*$, $t \to T$, and has little effect on optimal (ES, EW) or other summary statistics when $|\epsilon| \ll 1$. Further details about this stabilization term and its effects on both the HJB and NN framework will be discussed in Section 4.2. The objective function in (3.1.22) serves as the basis for the value function in the HJB framework and the loss function for the NN method.

Remark 3.1.2 (Induced time consistent policy)**.** *Note that a strategy based on* $(PCEE_{t_0}(\kappa))$ *is formally a pre-commitment strategy (i.e., not time consistent). However, we will assume that the retiree actually follows the induced time consistent strategy (Forsyth, 2020; 2022; Strub et al., 2019). This control is identical to the pre-commitment control at time zero. See Subsection 3.1.7 for more discussion of this subtle point. In the following, we will refer to the strategy determined by (3.1.22) as the EW-ES optimal control, with the understanding that this refers to the induced time consistent control at any time* $t_n > t_0$.

3.1.7 Induced Time Consistent Policy

In this section, we review the concept of time consistency and relate its relevance to the $PCEE_{t_0}(\kappa)$ problem, (3.1.22).

Consider the optimal control $\mathcal{P}_0^*$ for problem (3.1.22),

$$\mathcal{P}_0^*(X(t_i^-), t_i) \; ; \; i = 0, \ldots, M \; . \tag{3.1.23}$$

Equation (3.1.23) can be interpreted as the optimal control for any time $t_i \geq t_0$, as a function of the state variables $X(t)$, as computed at t_0.

Now consider if we were to solve the problem (3.1.22) starting at a later time $t_k, k > 0$. This optimal control starting at t_k is denoted by:

$$\mathcal{P}_k^*(X(t_i^-), t_i) \; ; \; i = k, \ldots, M\} \; . \tag{3.1.24}$$

In general, the solution of (3.1.22) computed at t_k is not equivalent to the solution computed t_0:

$$\mathcal{P}_k^*(X(t_i^-), t_i) \neq \mathcal{P}_0^*(X(t_i^-), t_i) \; ; \; i \geq k > 0. \tag{3.1.25}$$

This non-equivalence makes problem (3.1.22) *time inconsistent*, implying that the investor will have the incentive to deviate from the control computed at time t_0 at later times. This type of control is considered a *pre-commitment* control since the investor would need to commit to following the strategy at all times following t_0. Some authors describe pre-commitment controls as non-implementable because of the incentive to deviate.

In our case, however, the pre-commitment control from (3.1.22) can be shown to be identical to the time consistent control for an alternative version of the objective function. By holding W^* fixed at the optimal value (at time zero), we can define the time consistent equivalent problem (TCEQ). Noting that the inner supremum in (3.1.22) is a continuous function of W^*, we define the optimal value of W^* as

$$\mathcal{W}^*(s,b) \;=\; \arg\max_{W^*} \left\{ \sup_{\mathcal{P}_0 \in \mathcal{A}} \left\{ E_{\mathcal{P}_0}^{X_0^-, t_0^-} \left[\sum_{i=0}^{M} q_i \; + \; \kappa \left(W^* + \frac{1}{\alpha} \min(W_T - W^*, 0) \right) \middle| X(t_0^-) = (s,b) \right] \right\} \right\} . \tag{3.1.26}$$

With a given initial wealth of W_0^-, this gives the following result from Forsyth (2020):

Proposition 3.1.1 (Pre-commitment strategy equivalence to a time consistent policy for an alternative objective function)**.** *The pre-commitment EW-ES strategy found by solving $J\left(s,b,t_0^-\right)$ from (3.1.22), with fixed $W^* = \mathcal{W}^*$ from Equation 3.1.26, is identical to the time consistent strategy for the equivalent problem TCEQ (which has fixed $\mathcal{W}^*(0,W_0^-)$), with the following value function:*

$(TCEQ_{t_n}(\kappa/\alpha))$:

$$\tilde{J}\left(s,b,t_n^-\right) = \sup_{\mathcal{P}_n\in\mathcal{A}}\left\{E_{\mathcal{P}_n}^{X_n^-,t_n^-}\left[\sum_{i=n}^{M} q_i + \frac{\kappa}{\alpha}\min(W_T - \mathcal{W}^*(0,W_0^-),0)\middle| X(t_n^-) = (s,b)\right]\right\}. \tag{3.1.27}$$

Proof. This follows similar steps as in Forsyth (2020), proof of Proposition (6.2). □

With fixed W^*, $TCEQ_{t_n}(\kappa/\alpha)$ is based on a target-based shortfall as its measure of risk, which is trivially time consistent. W^* has the convenient interpretation of a disaster level of final wealth, as specified at time zero. Since the optimal controls for $PCEE_{t_0}(\kappa)$ and $TCEQ_{t_n}(\kappa/\alpha)$ are identical, we regard $TCEQ_{t_n}(\kappa/\alpha)$ as the EW-ES induced time consistent strategy (Strub et al., 2019), which is implementable since the investor will have no incentive to deviate from a strategy computed at t_0 at later times.

For further discussion concerning the relationship between pre-commitment, time consistent, and induced time consistent strategies, we refer the reader to Bjork and Murgoci (2010; 2014); Forsyth (2020); Strub et al. (2019); Vigna (2014; 2017).

3.2 HJB Dynamic Programming Optimization Framework

The HJB framework uses dynamic programming, creating sub-problems from each time step in the problem and moving backward in time. For the convenience of the reader, we will summarize the algorithm in Forsyth (2022) here.

3.2.1 Deriving Auxiliary Function from $PCEE_{t_0}(\kappa)$

The HJB framework begins with defining auxiliary functions based on the objective function (3.1.22) and the underlying stochastic processes. An equivalent problem is then formulated, which will then be solved to find the optimal value function.

We begin by interchanging the $\sup_{\mathcal{P}_0}$ and $\sup_{W^*}$ operators. This will serve as the starting point of the HJB solution.

$$J\left(s,b,t_0^-\right) = \sup_{W^*}\sup_{\mathcal{P}_0\in\mathcal{A}}\left\{E_{\mathcal{P}_0}^{X_0^-,t_0^-}\left[\sum_{n=0}^{M} q_n + \kappa\left(W^* + \frac{1}{\alpha}\min(W_T - W^*, 0)\right)\right.\right.$$
$$\left.\left. +\epsilon W_T \middle| X(t_0^-) = (s,b)\right]\right\} . \tag{3.2.1}$$

The auxiliary function which needs to be computed in the dynamic programming framework at each time t_n will have an associated strategy for any $t_n > 0$ that is equivalent with the solution of $PCEE_{t_0}(\kappa)$ for a fixed W^*. For a full discussion of pre-commitment and time-consistent ES strategies, we refer to the reader to Forsyth (2020), which also includes a proof with similar steps of how the following auxiliary function is derived from (3.2.1). Including W^* in the state space gives us the expanded state space $\hat{X} = (s,b,W^*)$. The auxiliary function $V(s,b,W^*,t) \in \Omega = [0,\infty)\times(-\infty,+\infty)\times(-\infty,+\infty)\times[0,\infty)$ is defined as,

$$V(s,b,W^*,t_n^-) = \sup_{\mathcal{P}_n\in\mathcal{A}_n}\left\{E_{\mathcal{P}_n}^{\hat{X}_n^-,t_n^-}\left[\sum_{n=0}^{M} q_n + \kappa\left(W^* + \frac{1}{\alpha}\min((W_T - W^*),0)\right)\right.\right.$$
$$\left.\left. +\epsilon W_T \middle| \hat{X}(t_n^-) = (s,b,W^*)\right]\right\} .$$

$$\text{subject to} \quad \begin{cases} (S_t, B_t) \text{ follow processes (3.1.3) and (3.1.4)}; \quad t \notin \mathcal{T} \\ W_n^+ = S_n^- + B_n^- - q_n\,;\ \hat{X}_n^+ = (S_n^+, B_n^+, W^*) \\ S_n^+ = p_n(\cdot)W_n^+\,;\ B_n^+ = (1-p_n(\cdot))W_n^+ \\ (q_n(\cdot), p_n(\cdot)) \in \mathcal{Z}(W_n^-, W_n^+, t_n) \\ i = n,\ldots,M\ ;\ t_n \in \mathcal{T} \end{cases} \tag{3.2.2}$$

3.2.2 Applying Dynamic Programming at Rebalancing Times

The principle of dynamic programming is applied at each $t_n \in \mathcal{T}$ on (3.2.2). As usual, the optimal control needs to be computed in reverse time order. We split the $\sup_{\mathcal{P}_n}$ operator into $\sup_{q\in\mathcal{Z}_q}\sup_{p\in\mathcal{Z}_p(w^- - q,t)}$.

$$
\begin{aligned}
V(s,b,W^*,t_n^-) &= \sup_{q\in\mathcal{Z}_q} \sup_{p\in\mathcal{Z}_p(w^- - q,t)} \left\{ q + \left[V((w^- - q)p, (w^- - q)(1-p), W^*, t_n^+) \right] \right\} \\
&= \sup_{q\in\mathcal{Z}_q} \left\{ q + \left[\sup_{p\in\mathcal{Z}_p(w^- - q,t)} V((w^- - q)p, (w^- - q)(1-p), W^*, t_n^+) \right] \right\} \\
& \qquad w^- = s + b \ . \qquad (3.2.3)
\end{aligned}
$$

Let $\overline{V}$ denote the upper semi-continuous envelope of V, which will have already been computed as the algorithm progresses backward through time. The optimal allocation control $p_n(w,W^*)$ at time t_n is determined from

$$
p_n(w,W^*) = \begin{cases} \arg\max_{p'\in[0,1]} \overline{V}(wp', w(1-p'), W^*, t_n^+), & w > 0 \ ; \ t_n \neq t_M \\ 0, & w \leq 0 \ \text{ or } t_n = t_M \end{cases} \ . \qquad (3.2.4)
$$

The control q is then determined from

$$
q_n(w,W^*) = \arg\max_{q'\in\mathcal{Z}_q} \left\{ q' + \overline{V}((w-q')p_n(w-q',W^*), (w-q')(1-p_n(w-q',W^*)), W^*, t_n^+) \right\} \ . \qquad (3.2.5)
$$

Using these controls for t_n, the solution is then advanced backwards across time from t_n^+ to t_n^- by

$$
\begin{aligned}
V(s,b,W^*,t_n^-) &= q_n(w^-,W^*) + \overline{V}(w^+ p_n(w^+,W^*), w^+(\ 1 - p_n(w^+,W^*)\ ), W^*, t_n^+) \\
& \qquad w^- = s + b \ ; \ w^+ = s + b - q_n(w^-,W^*) \ . \qquad (3.2.6)
\end{aligned}
$$

At $t = T$, we have the terminal condition

$$
V(s,b,W^*,T^+) = \kappa\left(W^* + \frac{\min((s+b-W^*),0)}{\alpha} \right) . \qquad (3.2.7)
$$

3.2.3 Conditional Expectations between Rebalancing Times

For $t \in (t_{n-1},t_n)$, there are no cash flows, discounting (all quantities are inflation-adjusted), or controls applied. Hence the tower property gives, for $0 < h < (t_n - t_{n-1})$,

$$V(s,b,W^*,t) \;=\; E\Big[V(S(t+h),B(t+h),W^*,t+h)\big|S(t)=s,B(t)=b\Big] \;;\; t \in (t_{n-1},t_n-h) \;. \tag{3.2.8}$$

To find this conditional expectation based on parametric models of the stock and bond processes, Itô's Lemma for jump processes (Tankov and Cont, 2009) is first applied in Equation (3.2.8), using Equations (3.1.3) and (3.1.4). This gives

$$\begin{aligned} V_t + \frac{(\sigma^s)^2 s^2}{2} V_{ss} + (\mu^s - \lambda^s_\xi \gamma^s_\xi) s V_s + \lambda^s_\xi \int_{-\infty}^{+\infty} V(e^y s,b,t) f^s(y)\, dy + \frac{(\sigma^b)^2 b^2}{2} V_{bb} \\ + (\mu^b + \mu^b_c \mathbf{1}_{\{b<0\}} - \lambda^b_\xi \gamma^b_\xi) b V_b + \lambda^b_\xi \int_{-\infty}^{+\infty} V(s,e^y b,t) f^b(y)\, dy - (\lambda^s_\xi + \lambda^b_\xi) V + \rho_{sb}\sigma^s\sigma^b s b V_{sb} = 0 \;, \\ s \geq 0 \;. \end{aligned} \tag{3.2.9}$$

where the density functions $f^s(y), f^b(y)$ are as given in equation (3.1.1).

In computational practice, this resulting PIDE is solved using Fourier methods discussed in Forsyth and Labahn (2019).

3.2.4 Equivalence with $PCEE_{t_0}(\kappa)$

Proceeding backward in time, the auxiliary function $V(s,b,W^*,t_0^-)$ is determined at time zero. Problem $PCEE_{t_0}(\kappa)$ is then solved using a final optimization step

$$J(s,b,t_0^-) = \sup_{W'} V(s,b,W',t_0^-) \;. \tag{3.2.10}$$

Notice that $V(s,b,W',t_0^-)$ denotes the auxiliary function for the beginning of the investment period, and represents the last step (going backward) in solving the dynamic programming formulation. To obtain this, we begin with Equation (3.2.7) and recursively work backwards in time; then we obtain Equation (3.1.22) by interchanging $\sup_{W'}\sup_{\mathcal{P}}$ in the final step.

This formulation (3.2.2-3.2.8) is equivalent to problem $PCEE_{t_0}(\kappa)$.

3.3 Computational Details: Hamilton-Jacobi-Bellman (HJB) PDE Framework

For a detailed description of the numerical algorithm used to solve the HJB equation framework described in Section 3.2, we refer the reader to Forsyth (2022). We summarize the method here.

First, we solve the auxiliary problem (3.2.2), with fixed values of W^*, κ and α. The state space in $s > 0$ and $b > 0$ is discretized using evenly spaced nodes in log space to create a grid to represent cases. A separate grid is created in a similar fashion to represent cases where wealth is negative. The Fourier methods discussed in Forsyth and Labahn (2019) are used to solve the PIDE representing market dynamics between rebalancing times. Both controls for withdrawal and allocation are discretized using equally spaced grids. The optimization problem (3.2.4) is solved first for the allocation control by exhaustive search, storing the optimal for each discretized wealth node. The withdrawal control in (3.2.5) can then be solved in a similar fashion, using the previously stored allocation control to evaluate the right-hand side of (3.2.5). Linear interpolation is used where necessary. The stored controls are used to advance the solution in (3.2.7).

Since the numerical method just described assumes a constant W^*, an outer optimization step to find the optimal W^* (candidate Value-at-Risk) is necessary. Given an approximate solution to (3.2.2) at $t = 0$, the full solution to $PCEE_{t_0}(\kappa)$ (3.1.22) is determined using Equation (3.2.10). A coarse grid is used at first for an exhaustive search. This is then used as the starting point for a one-dimensional optimization algorithm on finer grids.

3.4 Neural Network Formulation

As an alternative to the HJB framework, we develop a neural network framework to solve the stochastic optimal control problem (3.1.22), which has the following characteristics:

(i) The NN framework is data driven and does not require a parametric model of asset returns. This avoids explicitly postulating parametric stochastic processes and the estimation of associated parameters. In addition, this allows us to add auxiliary market signals/variables (although we do not exploit this idea in this work).

(ii) The NN framework avoids the computation of high-dimensional conditional expectations by solving for the control at all times directly from a single standard unconstrained optimization, instead of using dynamic programming (see Van Staden et al. (2023) for a discussion of this). Since the control is low-dimensional, the framework can exploit this to avoid the *curse of dimensionality* by solving for the control directly, instead of via value iteration such as in the HJB dynamic programming method (Van Staden et al., 2023). Such an approach also avoids backward error propagation through rebalancing times.

(iii) If the optimal control is a continuous function of time and state, the control approximated by the NN will reflect this property. If the optimal control is discontinuous, the NN approximation produces a smooth approximation. While not required by the original problem formulation in (3.1.22), this continuity property likely leads to practical benefits for an investment policy.

(iv) The NN method is further scalable in the sense that it could be easily adapted to problems with longer horizons or higher rebalancing frequency without significantly increasing the computational complexity of the problem. This is in contrast to existing approaches using a stacked neural network approach (Tsang and Wong, 2020).

We now formally describe the proposed NN framework and demonstrate aforementioned properties. We approximate the control in $\mathcal{P}$ directly by using feed-forward, fully-connected neural networks. Given parameters $\boldsymbol{\theta}_p$ and $\boldsymbol{\theta}_q$, i.e. NN weights and biases, $\hat{p}(W(t_n), t_n, \boldsymbol{\theta}_p)$ and $\hat{q}(W(t_n), t_n, \boldsymbol{\theta}_q)$ approximate the controls p_n and q_n respectively,

$$\begin{aligned}
\hat{q}(W_n^-, t_n^-, \boldsymbol{\theta}_q) &\simeq q_n(W_n^-) \ ; \ n = 0, \dots, M \\
\hat{p}(W_n^+, t_n^+, \boldsymbol{\theta}_p) &\simeq p_n(W_n^+) \ ; \ n = 0, \dots, M-1 \\
\hat{\mathcal{P}} &= \{(\hat{q}(\cdot), \hat{p}(\cdot))\} \simeq \mathcal{P}
\end{aligned}$$

The functions $\hat{p}$ and $\hat{q}$ take time as one of the inputs, and therefore we can use just two NN functions to approximate control $\mathcal{P}$ across time instead of defining a NN at each rebalancing time. In this section, we discuss how we solve problem (3.1.22) using this approximation and then provide a description of the NN architecture that is used. We discuss the precise formulation used by the NN, including activation functions that encode the stochastic constraints.

3.4.1 Neural Network Optimization for $PCEE_{t_0}(\kappa)$

We begin by describing the NN optimization problem based on the stochastic optimal control problem (3.1.22). We first recall that, in the formulation in Section 3.2, controls q_n and p_n are functions of wealth only. Our goal is to choose NN weights $\boldsymbol{\theta}_p$ and $\boldsymbol{\theta}_q$ by solving (3.1.22), with $\hat{q}(W_n^-,t_n^-,\boldsymbol{\theta}_q)$ and $\hat{p}(W_n^+,t_n^+,\boldsymbol{\theta}_p)$ approximating feasible controls $(q_n,p_n) \in \mathcal{Z}(W_n^-,W_n^+,t_n)$ for $t_n \in \mathcal{T}$. For an arbitrary set of controls $\hat{\mathcal{P}}$ and wealth level W^*, we define the NN performance criteria V_{NN} as

$$
\begin{aligned}
V_{NN}(\hat{\mathcal{P}},W^*,s,b,t_0^-) \;&=\; E_{\hat{\mathcal{P}}_0}^{X_0^-,t_0^-}\Bigg[\sum_{n=0}^{M}\hat{q}_n \;+\; \kappa\bigg(W^* + \frac{1}{\alpha}\min(W_T - W^*,0)\bigg) \\
&\qquad +\epsilon W_T \Big| X(t_0^-) = (s,b)\Bigg] \,.
\end{aligned}
$$

$$
\text{subject to } \begin{cases} (S_t,B_t) \text{ follow processes (3.1.3) and (3.1.4)}; \quad t \notin \mathcal{T} \\ W_n^+ = S_n^- + B_n^- - q_n\,;\; X_n^+ = (S_n^+,B_n^+) \\ S_n^+ = \hat{p}_n(\cdot)W_n^+\,;\; B_n^+ = (1-\hat{p}_n(\cdot))W_n^+ \\ (\hat{q}_n(\cdot),\hat{p}_n(\cdot)) \in \mathcal{Z}(W_n^-,W_n^+,t_n) \\ i = 0,\ldots,M\;;\; t_n \in \mathcal{T} \end{cases} \tag{3.4.1}
$$

The optimal value function J_{NN} (at t_0^-) is then given below,

$$
J_{NN}(s,b,t_0^-) = \sup_{W^*}\sup_{\hat{\mathcal{P}}\in\mathcal{A}} V_{NN}(\hat{\mathcal{P}},W^*,s,b,t_0^-)\;. \tag{3.4.2}
$$

Next we describe the structure of the neural networks and feasibility encoding.

3.4.2 Neural Network Framework

Consider two fully-connected feed-forward NNs, with $\hat{p}$ and $\hat{q}$ determined by parameter vectors $\boldsymbol{\theta}_p \in \mathbb{R}^{\nu_p}$ and $\boldsymbol{\theta}_q \in \mathbb{R}^{\nu_q}$ (representing NN weights and biases), respectively. The two NNs can differ in the choice of activation functions and in the number of hidden layers and nodes per layer. Each NN takes input of the same form $(W(t_n),t_n)$, but the withdrawal NN $\hat{q}$ takes the state variable observed before withdrawal, $(W(t_n^-),t_n)$, and the allocation NN $\hat{p}$ takes the state variable observed after withdrawal, $(W(t_n^+),t_n)$.

In order for the NN to generate a feasible control as specified in (3.4.5), we use a modified sigmoid activation function to scale the output from the withdrawal NN $\hat{q}$ according to the $PCEE_{t_0}(\kappa)$ problem's constraints on the withdrawal amount q_n, as given in Equation (3.1.10). This ultimately allows us to perform unconstrained optimization on the NN training parameters.

Specifically, assuming $x \in [0,1]$, the function $f(x) := a+(b-a)x$ scales the output to be in the range $[a,b]$. We restrict withdrawal to $\hat{q}$ in $[q_{\min}, q_{\max}]$. We note that this withdrawal range $q_{\max} - q_{\min}$ depends on wealth W^-, see from (3.1.10). Define the range of permitted withdrawal as follows,

$$\text{range} = \begin{cases} q_{\max} - q_{\min} \ ; & \text{if } W_n^- > q_{\max} \\ W^- - q_{\min} \ ; & \text{if } q_{\min} < W_n^- < q_{\max} \\ 0 \ ; & \text{if } W_n^- < q_{\min} \end{cases} .$$

More concisely, we have the following mathematical expression:

$$\text{range} = \max\big((\min(q_{\max}, W^-) - q_{\min}), 0\big) .$$

Let $z \in \mathbb{R}$ be the NN output before the final output layer of $\hat{q}$. Note that z depends on input features, state and time, before being transformed by the activation function. We then have the following expression for the final withdrawal,

$$\begin{aligned} \hat{q}(W^-, t, \boldsymbol{\theta_q}) &= q_{\min} + \text{range} \cdot \left(\frac{1}{1+e^{-z}}\right) \\ &= q_{\min} + \max\big((\min(q_{\max}, W^-) - q_{\min}), 0\big) \left(\frac{1}{1+e^{-z}}\right) . \end{aligned}$$

Note that the sigmoid function $\frac{1}{1+e^{-z}}$ is a mapping from $\mathbb{R} \to [0,1]$.

Similarly, we use a softmax activation function on the NN output of the $\hat{p}$, in order to impose no-shorting and no-leverage constraints:

$$Softmax\big((z_1, \ldots, z_K)\big) = \begin{bmatrix} \exp(z_1) \\ \vdots \\ \exp(z_K) \end{bmatrix} \cdot \frac{1}{\sum_{j=1}^{K} \exp(z_j)}, \tag{3.4.3}$$

where the elements of z are indexed by $j \in \{1, \ldots, K\}$ and each represent one of the K nodes in the output layer.

With these output activation functions, it can be easily verified that $(\hat{q}_n(\cdot), \hat{p}_n(\cdot)) \in \mathcal{Z}\left(W_n^-, W_n^+, t_n\right)$ always. Using defined NN, this transforms problem (3.4.2) of finding an optimal $\hat{\mathcal{P}}$ to solving the optimization problem below:

$$\begin{aligned}\hat{J}_{NN}(s,b,t_0^-) &= \sup_{W^*\in\mathbb{R}}\sup_{\boldsymbol{\theta_q}\in\mathbb{R}^{\nu_q}}\sup_{\boldsymbol{\theta_p}\in\mathbb{R}^{\nu_p}} \hat{V}_{NN}(\boldsymbol{\theta}_q,\boldsymbol{\theta}_p,W^*,s,b,t_0^-) \\ &= \sup_{(W^*,\boldsymbol{\theta_q},\boldsymbol{\theta_p})\in\mathbb{R}^{\nu_q+\nu_p+1}} \hat{V}_{NN}(\boldsymbol{\theta}_q,\boldsymbol{\theta}_p,W^*,s,b,t_0^-) \qquad (3.4.4)\end{aligned}$$

It is worth noting here that, while the original control $\mathcal{P}$ is constrained in (3.1.13), the formulation (3.4.4) is an unconstrained optimization over $\boldsymbol{\theta}_q$, $\boldsymbol{\theta}_p$, and W^*. Hence we can solve it directly using a standard gradient descent technique. In the numerical experiments detailed in Sections 4.2 and 4.3, we use Adam stochastic gradient descent (Kingma and Ba, 2014) to determine the optimal points $\boldsymbol{\theta}_{\boldsymbol{q}}^*$, $\boldsymbol{\theta}_{\boldsymbol{p}}^*$, and W^*.

The output of NN $\hat{q}$ yields the amount to withdraw, while the output of NN $\hat{p}$ produces asset allocation weights.

Figure 3.4.1 presents the proposed NN. We emphasize the following key aspects of this NN structure.

(i) Time is an *input* to both NNs in the framework. Therefore, the parameter vectors $\boldsymbol{\theta}_q$ and $\boldsymbol{\theta}_p$ are constant and do not vary with time.

(ii) At each rebalancing time, the wealth observation before withdrawal is used to construct the feature vector for $\hat{q}$. The resulting withdrawal is then used to calculate wealth after withdrawal, which is an input feature for $\hat{p}$.

(iii) Standard sigmoid activation functions are used at each *hidden layer* output.

(iv) The activation function for the withdrawal output is different from activation function for allocation. Control $\hat{q}$ uses a modified sigmoid function, which is chosen to transform its output according to (3.1.10). Control $\hat{p}$ uses a softmax activation which ensures that its output gives only positive weights for each portfolio asset that sum to one, as specified in (3.1.11). By constraining the NN output this way through proposed activation functions, we can use unconstrained optimization to train the NN.

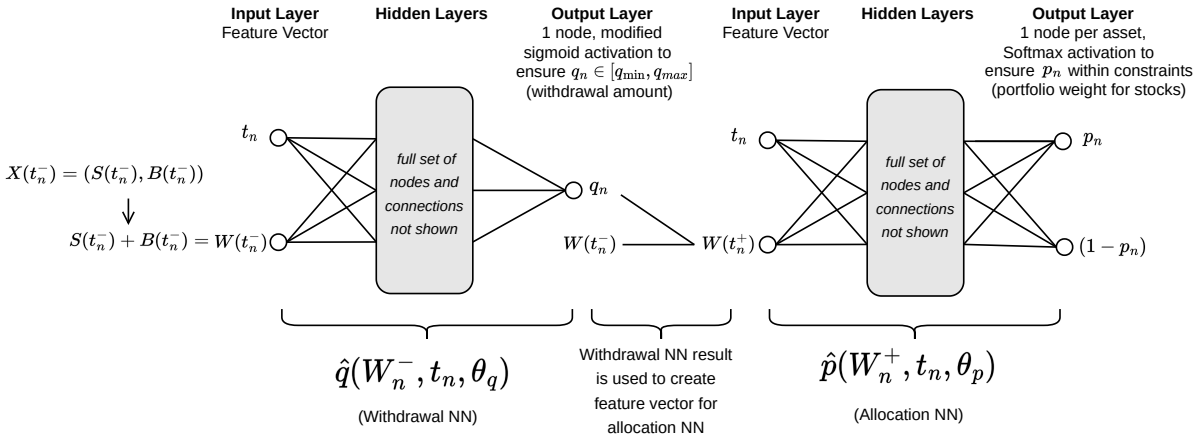


FIGURE 3.4.1: *Illustration of the NN framework as per Section 3.4.2. Additional technical details can be found in 3.5.*

3.4.3 NN Estimate of the Optimal Control

Now we describe the training optimization problem for the proposed data driven NN framework, which is agnostic to the underlying data generation process. We assume that a set of asset return trajectories are available, which are used to approximate the expectation in (3.4.1) for any given control. For NN training, we approximate the expectation in (3.4.1) based on a finite number of samples as follows:

$$\hat{V}_{NN}(\boldsymbol{\theta_q}, \boldsymbol{\theta_p}, W^*, s, b, t_0^-) =$$
$$\frac{1}{N}\sum_{j=1}^{N}\left[\sum_{n=0}^{M}\hat{q}((W_n)^j, t_n; \boldsymbol{\theta_q}) \; + \; \kappa\left(W^* + \frac{1}{\alpha}\min((W_T)^j - W^*, 0)\right) + \epsilon(W_T)^j \middle| X(t_0^-) = (s,b)\right]$$
$$\text{subject to} \begin{cases} ((S_t)^j, (B_t)^j) \text{ drawn from the } j^{th} \text{ sample of returns}; \quad t \notin \mathcal{T} \\ (W_n^+)^j = (S_n^-)^j + (B_n^-)^j - \hat{q}\left((W_{t_n}^-)^j, t_n, \boldsymbol{\theta_q}\right) \; ; \; (X_n^+)^j = (S_n^+, B_n^+)^j \\ (S_n^+)^j = \hat{p}\left((W_n^+)^j, t_n, \boldsymbol{\theta_p}\right) \; (W_n^+)^j \; ; \; (B_n^+)^j = (1 - \hat{p}\left((W_n^+)^j, t_n, \boldsymbol{\theta_p}\right)) \; (W_n^+)^j \\ (\hat{q}_n(\cdot), \hat{p}_n(\cdot)) \in \mathcal{Z}\left((W_n^-)^j, (W_n^+)^j, t_n\right) \\ n = 0, \ldots, M \; ; \; t_n \in \mathcal{T} \end{cases}, \tag{3.4.5}$$

where the superscript j represents the j^{th} path of joint asset returns and N is the total

number of sampled paths. For subsequent benchmark comparison, we generate price paths using processes (3.1.3) and (3.1.4). We are, however, agnostic as to the method used to generate these paths. We assume that random sample paths are independent. However, correlation between returns of different assets can be modelled. In addition, correlation between returns of different time periods can also be represented, e.g., block bootstrap resampling is designed to capture autocorrelation in the time series data.

The optimal parameters obtained by training the neural network are used to generate the control functions $\hat{q}^*(\cdot) := \hat{q}(\cdot; \boldsymbol{\theta}_{\boldsymbol{q}}^*)$ and $\hat{p}^*(\cdot) := \hat{p}(\cdot; \boldsymbol{\theta}_{\boldsymbol{p}}^*)$, respectively. With these functions, we can evaluate the performance of the generated control on testing data sets that are out-of-sample or out-of-distribution. We present the detailed results of such tests in Section 4.3.

3.5 Computational Details: NN Framework

3.5.1 NN Optimization

The NN framework, as described in Section 3.4 and illustrated in Figure 3.4.1, was implemented using the PyTorch library (Paszke et al., 2019). The withdrawal network $\hat{q}$, and allocation network $\hat{p}$ were both implemented with 2 hidden layers of 10 nodes each, with biases. Stochastic Gradient Descent (Ruder, 2016) was used in conjunction with the Adaptive Momentum optimization algorithm to train the NN framework (Kingma and Ba, 2014). The NN parameters and auxiliary training parameter W^* were trained with different initial learning rates. The same decay parameters and learning rate schedule were used. Weight decay (ℓ_2 penalty) was also employed to make training more stable. The training loop utilizes the auto-differentiation capabilities of the PyTorch library. Hyper-parameters used for NN training in this work's experiments are given in Table 3.5.1.

3.5.2 Transfer learning between different κ points

For high values of κ, the objective function is weighted more towards optimizing ES (lower risk). In these cases, optimal controls are more difficult to compute. This is because that the ES measure used (CVAR) is only affected by the sample paths below the 5^{th} percentile of terminal wealth, which are quite sparse. To overcome these training difficulties, we employ transfer learning (Tan et al., 2018) to improve training for the more difficult points on the efficient frontier. We begin training the model for the lowest κ from a random

initialization ('cold-start'), and then initialize the models for each increasing κ with the model for the previous κ. Through numerical experiments, we found this method made training far more stable and less likely to terminate in local minima for higher values of κ.

3.5.3 Running minimum tracking

The training loop tracks the minimum loss function value as training progresses and selects the model that had given the optimal loss function value based on the entire training dataset by the end of the specified number of training epochs.

NN framework hyper-parameter	Value
Hidden layers per network	2
number of nodes per hidden layer	10
Nodes have biases	True
number of iterations (#itn)	50,000
SGD mini-batch size	1,000
N, number of training paths	2.56×10^6
Optimizer	Adaptive Momentum
Initial Adam learning rate for $(\boldsymbol{\theta}_q, \boldsymbol{\theta}_p)$	0.5
Initial Adam learning rate for W^*	0.4
Adam learning rate decay schedule	$[0.70 \times \#\text{itn}, 0.97 \times \#\text{itn}]$, $\gamma = 0.20$
Adam β_1	0.9
Adam β_2	0.998
Adam weight decay (L2 Penalty)	0.0001
Transfer Learning between κ points	True
Take running minimum as result	True

TABLE 3.5.1: *Hyper-parameters used in training the NN framework for numerical experiments presented in this work.*

3.5.4 Standardization

To improve learning for the neural network, we normalize the input wealth using means and standard deviations of wealth samples from a reference strategy. We use the constant withdrawal and allocation strategy defined in Forsyth (2022) as the reference strategy

with $N = 2.56 \times 10^5$ simulated paths. Let W_t^b denote the wealth vector at time t based on simulations. Then $\bar{W}_t^b$ and $\sigma(W_t^b)$ denote the associated average wealth and standard deviation. Then we normalize the feature input to the neural network in the following way:

$$\tilde{W}_t = \frac{W_t - \bar{W}_t^b}{\sigma(W_t^b)}$$

For the purpose of training the neural network, the values $\bar{W}_t^b$ and $\sigma(W_t^b)$ are just constants, and we can use any reasonable values. This input feature normalization is done for both withdrawal and allocation NNs.

In Section 4.3, we show out-of-sample and out-of-distribution tests that $\bar{W}_t^b$ and $\sigma(W_t^b)$ do not need to be related to the testing data as long as these are reasonable values. In Section 3.4, when referring to W as part of the input to the NN functions $\hat{q}$ and $\hat{p}$, we use the standardized $\tilde{W}$ for computation.

Chapter 4

Computational Results: HJB and NN Frameworks

4.1 Data

For the computational study in this chapter, we use data from the Center for Research in Security Prices (CRSP) on a monthly basis over the 1926:1-2019:12 period.[1] The specific indices used are the CRSP 10-year U.S. Treasury index for the bond asset[2] and the CRSP value-weighted total return index for the stock asset[3]. All of these various indexes are in nominal terms, so we adjust them for inflation by using the U.S. CPI index, also supplied by CRSP. We use real indexes since investors funding retirement spending should be focused on real (not nominal) wealth goals.

We use above market data in two different ways in subsequent investigations:

(i) *Stochastic model calibration:* Any data set referred to in this work as *synthetic data* is generated by parametric stochastic models (SDEs) (as described in Section 3.1.2), whose parameters are calibrated to the aforementioned CRSP data by using the

[1]More specifically, results presented here were calculated based on data from Historical Indexes, ©2020 Center for Research in Security Prices (CRSP), The University of Chicago Booth School of Business. Wharton Research Data Services was used in preparing this article. This service and the data available thereon constitute valuable intellectual property and trade secrets of WRDS and/or its third-party suppliers.

[2]The 10-year Treasury index was calculated using monthly returns from CRSP dating back to 1941. The data for 1926-1941 were interpolated from annual returns in Homer and Sylla (2005).

[3]The stock index includes all distributions for all domestic stocks trading on major U.S. exchanges.

threshold technique (Cont and Mancini, 2011; Dang and Forsyth, 2016; Mancini, 2009). The data is inflation-adjusted so that all parameters reflect real returns. Table 4.1.1 shows the calibrated model parameters for processes (3.1.3) and (3.1.4), from Forsyth (2022) using historical CRSP data. The correlation ρ_{sb} is computed by removing any returns which occur at times corresponding to jumps in either series. See Dang and Forsyth (2016) for details of the technique for detecting jumps.

Calibrated Model Parameters

CRSP	μ^s	σ^s	λ^s	u^s	η_1^s	η_2^s	ρ_{sb}
	0.0877	0.1459	0.3191	0.2333	4.3608	5.504	0.04554
10-year Treasury	μ^b	σ^b	λ^b	u^b	η_1^b	η_2^b	ρ_{sb}
	0.0239	0.0538	0.3830	0.6111	16.19	17.27	0.04554

TABLE 4.1.1: *Estimated annualized parameters for double exponential jump diffusion model. Value-weighted CRSP index, 10-year US treasury index deflated by the CPI. Sample period 1926:1 to 2019:12.*

(ii) *Bootstrap resampling:* Any data set referred to in this work as *historical data* is generated by using the stationary block bootstrap method (Dichtl et al., 2016; Patton et al., 2009; Politis and Romano, 1994; Politis and White, 2004) to resample the historical CRSP data set. This method involves repeatedly drawing random sample block of random block size, with replacement, from the original data set, where the block size follows a geometric distribution with a given expected block size. To preserve correlation between asset returns, we use a paired sampling approach to simultaneously draw returns from both time series. This, in effect, shuffles the original data and can be repeated to obtain however many resampled paths one desires. Since the order of returns in the sequence is unchanged within the sampled block, this method accounts for some possible serial correlation in market data. Detailed pseudo-code for this method of block bootstrap resampling is given in Forsyth and Vetzal (2019).

We note that block resampling is commonly used by practitioners and academics (see for example Anarkulova et al. (2022); Cogneau and Zakamouline (2013); Dichtl et al. (2016); Scott and Cavaglia (2017); Simonian and Martirosyan (2022)). It will be used to carry out robustness check in Section 4.3. Note that for any reasonable

number of samples, the probability of repeating a resampled path is negligible (Ni et al., 2022b).

One important parameter for the block resampling method is the expected block size. Forsyth (2022) determines that a reasonable expected block size for paired resampling is about 0.25 years. The algorithm presented in Patton et al. (2009) is used to determine the optimal expected block size for the bond and stock returns separately; see Table 4.1.2.

Optimal expected block size for bootstrap resampling historical data

Data series	Optimal expected block size $\hat{b}$ (months)
Real 10-year Treasury index	4.2
Real CRSP value-weighted index	3.1

TABLE 4.1.2: *Optimal expected blocksize $\hat{b} = 1/v$ when the blocksize follows a geometric distribution $Pr(b = k) = (1 - v)^{k-1}v$. The algorithm in Patton et al. (2009) is used to determine $\hat{b}$. Historical data range 1926:1-2019:12.*

To train the neural networks, we require that the number of sampled paths, N, be sufficiently large to fully represent the underlying market dynamics. Subsequently, we first generate training data through Monte Carlo simulations of the parametric models described in (3.1.3) and (3.1.4). We emphasize however that in the proposed data driven NN framework, we only require return trajectories of the underlying assets. In later sections, we present results from NN trained on data from nonparametricaly generated, e.g., resampled historical data. We also demonstrate NN framework's robustness on test data.

4.2 Computational Results

We now present and compare performance of the optimal control from the HJB PDE and NN method respectively on synthetic data, with investment specifications given in Table 4.2.1. Each strategy's performance is measured w.r.t. to the objective function in (3.1.22), which is a weighted reward (EW) and risk (ES) measure. To trace out an efficient frontier in the (EW,ES) plane, we vary κ (the curve represents the (EW,ES) performance on a set of optimal Pareto points).

We first present strategies computed from the HJB framework described in Section 3. We verify that the numerical solutions are sufficiently accurate, so that this solution can be regarded as ground truth. We then present strategies computed using the NN framework of Section 3.4, and demonstrate their accuracy by comparing them to the ground truth computed from the HJB equation. We carry out further analysis by selecting an *interesting* point on the (EW,ES) efficient frontier, in particular $\kappa = 1.0$, to study in greater detail. The point $\kappa = 1.0$ is at the *knee* of the efficient frontier, which makes it desirable in terms of risk-reward tradeoff (picking the exact κ will be a matter of investor preference, however). This notion of the knee point is loosely based on the concept of a *compromise solution* of multi-objective optimization problems, which selects the point on the efficient frontier with the minimum distance to an unattainable ideal point (Marler and Arora, 2004). For this knee point of $\kappa = 1.0$, we analyze the controls and wealth outcomes under both frameworks. We also discuss some key differences between the HJB and NN frameworks' results and their implications.

Investment horizon T (years)	30
Equity market index	CRSP Cap-weighted index (real)
Bond index	10-year Treasury (US) (real)
Initial portfolio value W_0	1000
Cash withdrawal times	$t = 0,1,\ldots,30$
Withdrawal range	$[35, 60]$
Equity fraction range	[0,1]
Borrowing spread μ_c^b	0.0
Rebalancing interval (years)	1
Market parameters	See Figure 4.1.1
ε (stabilization term)	10^{-6}

TABLE 4.2.1: *Problem setup and input data. Monetary units: thousands of dollars.*

4.2.1 Strategies Computed from HJB Equation

We carry out a convergence test for the HJB framework by tracing the efficient frontier (i.e. varying the scalarization parameter κ) for solutions of varying refinement levels (i.e. number of grid points in the (s,b) directions). Figure 4.2.1 shows these efficient frontiers. As the efficient frontiers from various grid sizes all practically overlap each other, this demonstrates convergence of solutions computed from solving HJB equations.

Table 4.2.2 shows a detailed convergence test for a single point on the (EW, ES) frontier, using the PIDE method. The controls are computed using the HJB PDE, and stored. The stored controls are then used in Monte Carlo simulations, which are used to verify the PDE solution, and also generate various statistics of interest.

Algorithm in Section 3.2					Monte Carlo	
Grid	ES (5%)	$E[\sum_i q_i]/(M+1)$	Value Function	W^*	ES (5%)	$E[\sum_i q_i]/(M+1)$
512×512	-51.302	52.056	1.562430e+3	50.10	-45.936	52.07
1024×1024	-46.239	52.049	1.567299e+3	52.47	-45.102	52.05
2048×2048	-42.594	51.976	1.568671e+3	58.00	-42.623	51.97
4096×4096	-40.879	51.932	1.569025e+3	61.08	-41.250	51.93

TABLE 4.2.2: *HJB equation convergence test, real stock index: deflated real capitalization weighted CRSP, real bond index: deflated ten year treasuries. Scenario in Table 4.2.1. Parameters in Table 4.1.1. The Monte Carlo method used* 2.56×10^6 *simulations.* $\kappa = 1.0, \alpha = .05$. *Grid refers to the grid used in the Algorithm in Section 3.2:* $n_x \times n_b$, *where* n_x *is the number of nodes in the* $\log s$ *direction, and* n_b *is the number of nodes in the* $\log b$ *direction. Units: thousands of dollars (real).* $(M+1)$ *is the total number of withdrawals.* M *is the number of rebalancing dates.* $q_{\min} = 35.0$. $q_{\max} = 60$. *Algorithm in Section 3.2.*

The convergence is roughly first-order. This convergence test justifies the use of the HJB framework results as a ground-truth.

Remark 4.2.1 (Effect of Stabilization Term ϵW_T)**.** *Recall the stabilization term,* ϵW_T, *introduced in (3.1.22). We now provide motivation for its inclusion, and observe its effect on the control* $\hat{\mathcal{P}}$. *When* $W_t \gg W^*$ *and* $t \to T$, *the control will only weakly affect the objective function. This is because, in this situation,* $Pr[W_T < W^*] \simeq 0$ *and thus the allocation control will have little effect on the ES term in the objective (recall that* W^* *is held constant for the induced time consistent strategy, see section 3.1.7). In addition, the withdrawal is capped at* $q_{\max}$ *for very high values of* W_t, *so the withdrawal control does not depend on* W_t *in this case either. The stabilization term can be used to alleviate ill-posedness of the problem in this region.*

In Figure 4.2.2, we present the heat map of the allocation control computed from the HJB framework. Subplot (a) presents allocation control heat map for a small positive stabilization parameter $\epsilon = 10^{-6}$, while Subplot (b) presents allocation control heat map

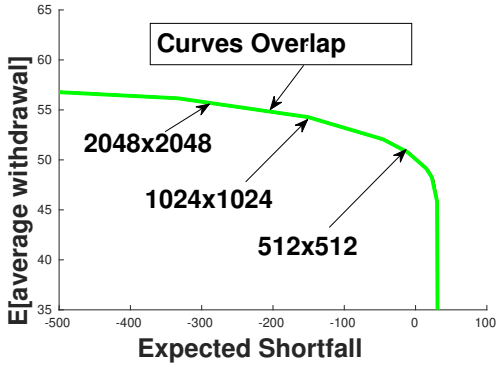


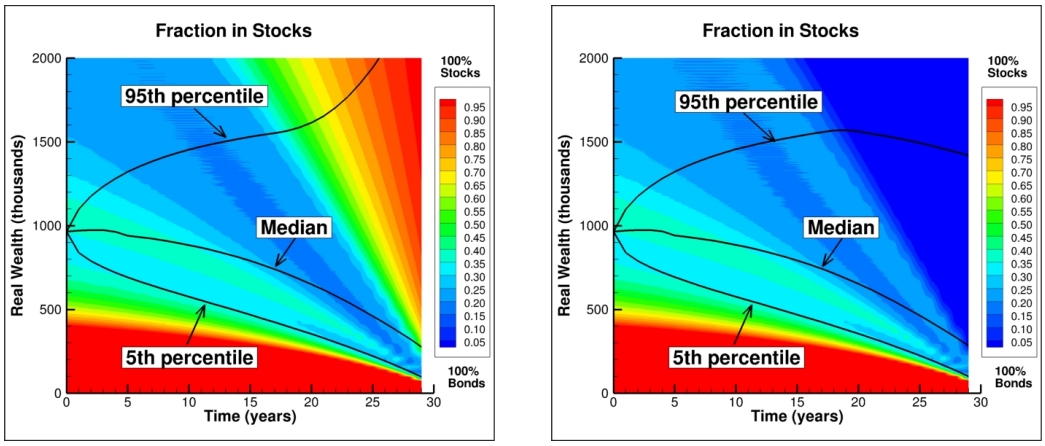
FIGURE 4.2.1: *EW-ES frontier, computed from problem (3.1.22). Note: Scenario in Table 4.2.1. Comparison of HJB solution performance with varying grid sizes. HJB solution performance computed on* 2.56×10^6 *observations of synthetic data. Parameters for synthetic data based on cap-weighted real CRSP, real 10 year treasuries (see Table 4.1.1).* $q_{min} = 35, q_{\max} = 60$. $\epsilon = 10^{-6}$. *Units: thousands of dollars.*

with $\epsilon = -10^{-6}$. In the ill-posed region (the top right region of the heat maps), the presence of ϵW_T, with $\epsilon = 10^{-6}$, forces the control to invest 100% in stocks to generate high terminal wealth. Conversely, changing the stabilization parameter to be negative ($\epsilon = -10^{-6}$) forces the control to invest completely in bonds.

We observe that the control behaves differently only at high level of wealth as $t \to T$ in both cases. The 5th and the 50th percentiles of control on the synthetic data set behave similarly in both the positive and negative ϵ cases. The 95th percentile curve tends towards higher wealth during later phases of the investment period when the ϵ is positive (Figure 4.2.2(a)), whereas the curve tends downward when ϵ is negative (Figure 4.2.2(b)). When the magnitude of ϵ is sufficiently small, its inclusion of ϵW_T in the objective function does not change summary statistics (to four decimal places when $|\epsilon| = 10^{-6}$). While the choice of negative or positive ϵ with small magnitude can to different allocation control scenarios at hight wealth level near the end of time horizon, the choice makes little difference from the perspective of the problem $PCEE_{t_0}(\kappa)$. If the investor reaches very high wealth near T, the choice between 100% stocks and 100% bonds does not matter as the investor always ends with $W_T \gg W^*$. Our experiments show that the control q is unaffected when the magnitude of ϵ is small and continues to call for maximum withdrawals at high levels of wealth as $t \to T$, just as described in Remark 4.2.1.

Comparing the optimal withdrawal strategy determined by solving stochastic optimal control problem (3.4.5) with a fixed withdrawal strategy (both strategies with dynamic asset allocation), Forsyth (2022) finds that the stochastic optimal strategy (3.4.5) is much

more efficient in withdrawing cash over the investment horizon. Accepting a very small amount of additional risk, the retiree can dramatically increase total withdrawals. For a more detailed discussion of the optimal control, we refer the reader to Forsyth (2022).



(a) *Fraction in stocks, HJB Control,* $\epsilon = 10^{-6}$

(b) *Fraction in stocks, HJB Control,* $\epsilon = -10^{-6}$

FIGURE 4.2.2: *Effect of* ϵ*: fraction in stocks computed from the problem (3.1.22). Note: investment setup is as in Table 4.2.1. HJB solution performance computed on* 2.56×10^6 *observations of synthetic data. Parameters for synthetic data based on cap-weighted real CRSP, real 10 year treasuries (see Table 4.1.1).* $q_{min} = 35, q_{\max} = 60$, $\kappa = 1.0$. $W^* = 58.0$ *for PIDE results. (a)* $\epsilon = 10^{-6}$. *(b)* $\epsilon = -10^{-6}$. *Units: thousands of dollars.*

4.2.2 Accuracy of Strategy Computed from NN framework

We compute the NN control following the framework discussed in Section 3.4. We compare the efficient frontiers obtained from the HJB equation solution, and the NN solution. From Figure 4.2.3, the NN control efficient frontier is almost indistinguishable from the HJB control efficient frontier. Detailed summary statistics for each computed point on the frontier can be found in Table 4.2.4, and a comparison of objective function values, for the NN and HJB control at each frontier point are discussed in subsection 4.2.5. For most points on the frontier, the difference in objective function values, from NN and HJB, is less than 0.1%. This demonstrates that the accuracy of the NN framework approximation of the ground-truth solution is more than adequate, considering that the difference between the NN solution and the PDE solution is about the same as the estimated PDE error (see Table 4.2.2).

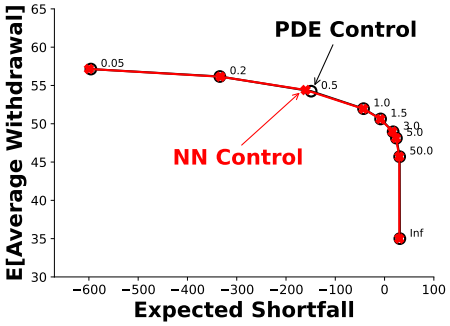


FIGURE 4.2.3: *Comparison of EW-ES frontier for the Neural Network (NN) and Hamilton-Jacobi-Bellman (HJB) Partial Differential Equation (PDE) methods, computed from the problem (3.1.22). Note: investment setup in Table 4.2.1. HJB solution performance computed on* 2.56×10^6 *observations of synthetic data. Parameters for synthetic data based on cap-weighted real CRSP, real 10 year treasuries (see Table 4.1.1). Control computed from the NN model, trained on* 2.56×10^6 *observations of synthetic data.* $q_{min} = 35, q_{\max} = 60$. $\epsilon = 10^{-6}$. *Units: thousands of dollars. Labels on nodes indicate* κ *parameter.*

4.2.3 Detailed efficient frontier comparisons

Table 4.2.3 shows the detailed efficient frontier, computed using the HJB equation method, using the 2048×2048 grid. Table 4.2.4 shows the efficient frontier computed from the NN framework. This should be compared to Table 4.2.3. Table 4.2.5 compares the objective function values, at various points on the efficient frontier, for the HJB and NN frameworks.

4.2.4 NN-approximated control

We now further analyze the control $\hat{\mathcal{P}}$ produced by the NN framework for $\kappa = 1$. Comparing Figure 4.2.4(b) with Figure 4.2.4(d), we observe that the withdrawal control $\hat{q}$ produced by the NN is practically identical to that produced by the HJB framework. However, there are differences in the allocation control heat maps. The NN heat map for allocation control p (Figure 4.2.4(a)) appears most similar to that of the HJB allocation heat map for negative ϵ (Figure 4.2.2(b)), but it is clear that the NN allocation heat map differs significantly from the HJB heat map for positive ϵ (Figure 4.2.2(a)) at high level of wealth as $t \to T$. The NN allocation control behaves differently from the HJB controls in this region, choosing a mix of stocks and bonds instead of choosing a 100% allocation in a single asset. Noting this difference is only at higher level of wealth near T, we see that the 5th percentile and

Detailed Efficient Frontier: HJB Framework

κ	ES (5%)	$E[\sum_i q_i]/(M+1)$	$Median[W_T]$
0.05	-596.00	57.14	124.36
0.2	-334.29	56.17	92.99
0.5	-148.99	54.25	111.20
1.0	-42.62	51.97	227.84
1.5	-8.05	50.63	298.20
3.0	17.42	48.95	380.36
5.0	24.09	48.12	414.60
50.0	30.60	45.70	519.03
∞	31.00	35.00	1003.47

TABLE 4.2.3: *Synthetic market results for HJB framework optimal strategies. Gives the detailed results used to construct HJB efficient frontier in Figure 4.2.3. Assumes the scenario given in Table 4.2.1. Stock index: real capitalization weighted CRSP stocks; bond index: ten year treasuries. Parameters from Table 4.1.1. Units: thousands of dollars. Statistics based on* 2.56×10^6 *Monte Carlo simulation runs. Control is computed using the Algorithm in Section 3.2, (*2048×2048 *grid) stored, and then used in the Monte Carlo simulations.* $q_{\min} = 35.0$, $q_{\max} = 60$. $(M+1)$ *is the number of withdrawals.* M *is the number of rebalancing dates.* $\epsilon = 10^{-6}$.

the median wealth curves are indistinguishable. The NN control's 95th percentile curve, however, is different and indeed the curve is in between the 95th percentile curves from the negative and positive versions of the HJB-generated control.

Drawing from this, we attribute the NN framework's inability to fully replicate the HJB control to the ill-posedness of the optimal control problem in the (top-right) region of high wealth level near T. The small value of ϵ means that the stabilization term contributes a very small fraction of the objective function value and thus has a very small gradient, relative to the first two terms in the objective function. Since we use stochastic gradient descent for optimization, we see a very small impact of ϵ. Moreover, the data for high levels of wealth as $t \to T$ is very sparse and so the effect of the small gradient is further reduced. As a result, the NN appears to smoothly extrapolate in this region and therefore avoids investment into a single asset. Recall that in Section 4.2.1, we stated that the choice in the signs of ϵ, with small ϵ, in the stabilization term is somewhat arbitrary and does not affect summary statistics. Therefore, we see that the controls produced by the two methods only differ irrelevant aspects, at least based on the EW and ES reward-risk consideration.

Detailed Efficient Frontier: NN Framework

κ	ES (5%)	$E[\sum_i q_i]/(M+1)$	$Median[W_T]$
0.05	-599.81	57.15	106.23
0.2	-333.01	56.14	78.59
0.5	-160.14	54.40	105.05
1	-43.02	51.95	227.79
1.5	-8.57	50.62	302.17
3	16.01	48.99	374.43
5	23.20	48.13	425.13
50	29.88	45.72	493.41
∞	29.90	35.00	947.60

TABLE 4.2.4: *Synthetic market results for NN framework optimal strategies. Gives the detailed results used to construct NN efficient frontier in Figure 4.2.3. Assumes the scenario given in Table 4.2.1. Stock index: real capitalization weighted CRSP stocks; bond index: ten year treasuries. Parameters from Table 4.1.1. Units: thousands of dollars. Training performance statistics based on* 2.56×10^5 *Monte Carlo simulation runs. Control is computed using the algorithm in Section 3.4.* $q_{\min} = 35.0$, $q_{\max} = 60$. $(M+1)$ *is the number of withdrawals.* M *is the number of rebalancing dates.* $\epsilon = 10^{-6}$.

It is interesting to observe that the proposed neural network framework is able to produce the *bang-bang* withdrawal control computed in Forsyth (2022), especially since we are using the continuous function $\hat{q}$ as an approximation.[4] A *bang-bang* control switches abruptly as shown here: the optimal strategy is to withdraw the minimum if the wealth is below a threshold, or else withdraw the maximum. As expected, the control threshold decreases as we move forward in time. We can see that the NN and HJB withdrawal controls behave very similarly at the 95th, 50th, and 5th percentiles of wealth (Figures 4.2.5(c) and 4.2.5(f)). Essentially, the optimal strategy withdraws at either $q_{\max}$ or $q_{\min}$, with a very small transition zone. This is in line with our expectations. By withdrawing less and investing more initially, the individual decreases the chance of running out of savings.

We also note that the NN allocation control presents a small spread between the 5th and 95th percentile of the fraction in stocks (Figure 4.2.5(a)). In fact, the maximum stock allocation for the 95th percentile never exceeds 40%, indicating that this is a stable low-risk strategy, which as we shall see, outperforms the Bengen (1994) strategy.

[4]Note that Forsyth (2022) shows that that in the continuous withdrawal limit, the the withdrawal control is bang-bang.

Objective Function Value Comparison: HJB Framework vs. NN Framework

κ	HJB equation	NN	% difference
0.05	1741.54	1741.71	0.01%
0.2	1674.41	1673.81	-0.04%
0.5	1607.26	1606.44	-0.05%
1	1568.45	1567.34	-0.07%
1.5	1557.46	1556.22	-0.08%
3	1569.71	1566.86	-0.18%
5	1612.16	1607.86	-0.27%
50	2946.70	2911.10	-1.21%

TABLE 4.2.5: *Objective function value comparison for the HJB equation and NN framework model results on range of κ values. Objective function values for both frameworks computed according to $PCEE_{t0}(\kappa)$ (higher is better). Assuming the scenario given in Table 4.2.1. Stock index: real capitalization weighted CRSP stocks; bond index: ten year treasuries. Parameters from Table 4.1.1. HJB solution statistics based on 2.56×10^6 Monte Carlo simulation runs. HJB control is computed using the Algorithm in Section 3.2, (2048×2048 grid) stored, and then used in the Monte Carlo simulations. NN Training performance statistics based on 2.56×10^5 Monte Carlo simulation runs. Control is computed using the NN framework in Section 3.4. $q_{\min} = 35.0$, $q_{\max} = 60$. $(M+1)$ is the number of withdrawals. M is the number of rebalancing dates. $\epsilon = 10^{-6}$.*

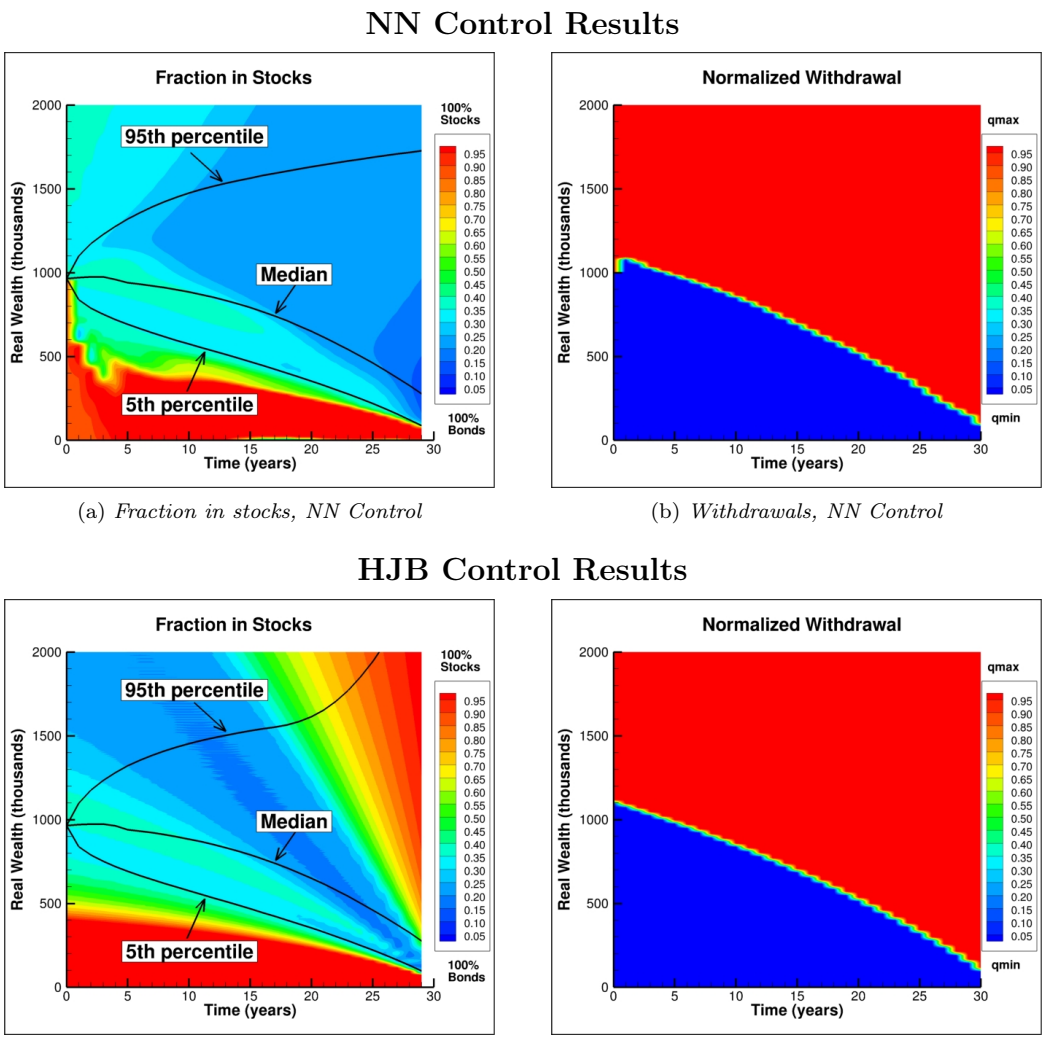
4.3 NN Model Robustness

A common pitfall of neural networks is over-fitting to the training data. Neural networks that are over-fitted do not have the ability to generalize to previously unseen data. Since future asset return paths cannot be predicted, it is important to ascertain that the computed strategy is not overfitted to the training data and can perform well on unseen return paths. In this section, we demonstrate the robustness of the NN model's generated controls.

We conduct three types of robustness tests: (i) out-of-sample testing, (ii) out-of-distribution testing, and (iii) control sensitivity to training distribution.

4.3.1 Out-of-sample testing

Out-of-sample tests involve testing model performance on an unseen data set sampled from the same distribution. In our case, this means training the NN on one set of SDE paths sampled from the parametric model, and testing on another set of paths generated using



(a) *Fraction in stocks, NN Control*

(b) *Withdrawals, NN Control*

(c) *Fraction in stocks, HJB Control*

(d) *Withdrawals, HJB Control*

FIGURE 4.2.4: *Heat map of controls: fraction in stocks and withdrawals, computed from the problem (3.1.22). Note: problem setup described in Table 4.2.1. HJB solution performance computed on* 2.56×10^6 *observations of synthetic data. Parameters for synthetic data based on cap-weighted real CRSP, real 10 year treasuries (see Table 4.1.1). NN model trained on* 2.56×10^6 *observations of synthetic data.* $q_{min} = 35, q_{\max} = 60$, $\kappa = 1.0$. $W^* = 59.1$ *for NN results.* $W^* = 58.0$ *for the HJB results.* $\epsilon = 10^{-6}$. *Normalized withdrawal* $(q - q_{\min})/(q_{\max} - q_{\min})$. *Units: thousands of dollars.*

a different random seed. We present the efficient frontier generated by computed controls

on this new data set in Figure 4.3.1, which shows almost unchanged performance on the out-of-sample test set.

4.3.2 Out-of-distribution testing

Out-of-distribution testing involves evaluating the performance of computed control on an entirely new data set sampled from a different distribution. Specifically test data is not generated from the parametric model used to produce training data, but is instead bootstrap resampled from historical market returns via the method described in Section 4.1. We vary the expected block sizes to generate multiple testing data sets of 2.56×10^5 paths.

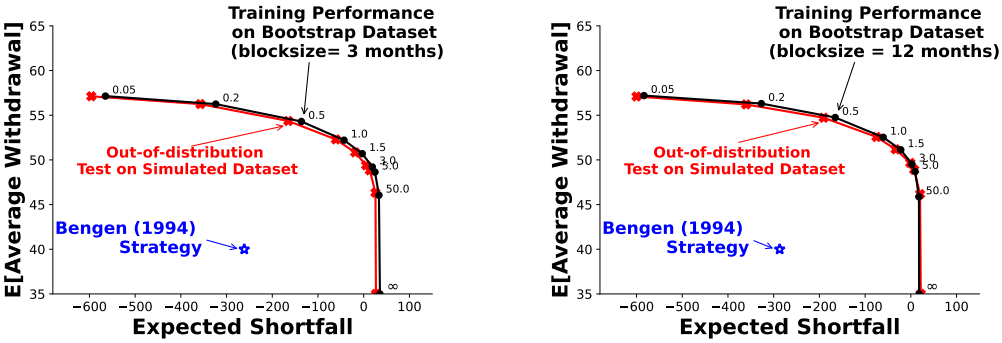
In Figure 4.3.2, we see that for each block size tested, the efficient frontiers are fairly close, indicating that the controls are relatively robust. Note that the efficient frontiers for test performance in the historical market with expected block size of 1 and 3 months plot slightly above the synthetic market frontier. We conjecture that this may be due to more pessimistic tail events in the synthetic market.

The out-of-sample and out-of-distribution tests verify that the neural network is not over-fitting to the training data, and is generating an effective strategy, at least based on our block resampling data.

4.3.3 Control sensitivity to training distribution

To further test the NN framework's adaptability to other training data sets, we train the NN framework on historical data (with expected block sizes of both 3 months and 12 months) and then test the resulting control on synthetic data. In Figure 4.3.3, we compare the training performance and the test performance. The EW-ES frontiers for the test results on the synthetic data are very close to the results on the bootstrap market data (training data set). This shows the NN framework's adaptability to use alternative data sets to learn, with the added advantage of not being reliant on a parametric model, which is prone to miscalibration. Figure 4.3.3 also shows that, in all cases, in the synthetic or historical market, the EW-ES control significantly outperforms the Bengen *4% Rule* [5] (Bengen, 1994).

[5]The results for the Bengen strategy on the historical test data were computed with fixed 4% withdrawals and constant allocation of 30% in stocks for expected block size of 3 months, and 35% in stocks for expected block size of 12 months. These were found to be the best performing constant allocations when paired with constant 4% withdrawals, in terms of ES efficiency.

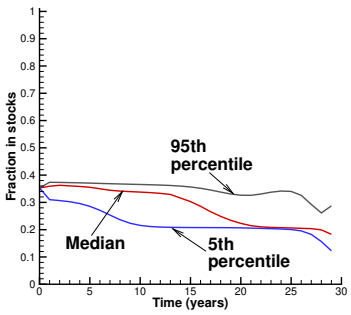


(a) *Historical training data, block size = 3 months*

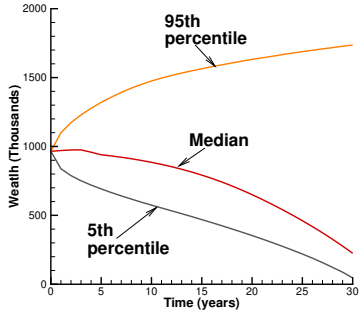
(b) *Historical training data, block size = 12 months*

FIGURE 4.3.3: *Training on historical data. EW-ES frontiers of controls generated by NN model trained on* 2.56×10^5 *observations of historical data with expected block sizes of a) 3 months and b) 12 months, each tested on* 2.56×10^5 *observations of synthetic data. Parameters based on real CRSP index and real 10-year U.S. Treasuries (see Table 4.1.1). Historical data in range 1926:1-2019:12. Units: thousands of dollars.* $q_{min} = 35; q_{max} = 60$. *The Bengen (1994) results are based on bootstrap resampling of the historical data. Labels on nodes indicate* κ *parameter values. Simulated testing data refers to Monte Carlo simulations using the SDEs* (3.1.3) *and* (3.1.4).

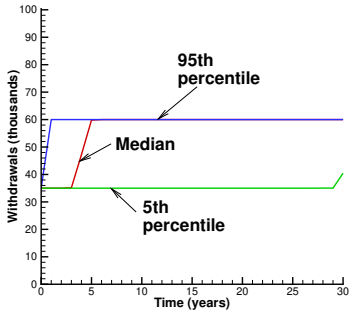
NN Control Results



(a) *Percentiles fraction in stocks, NN Control,* $\epsilon = 10^{-6}$

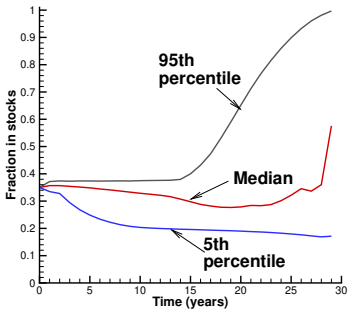


(b) *Percentiles wealth, NN Control,* $\epsilon = 10^{-6}$

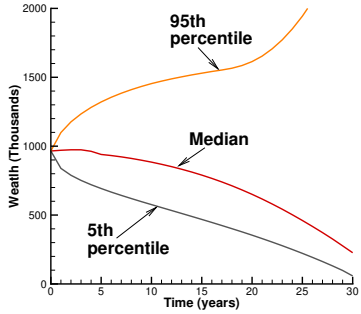


(c) *Percentiles withdrawals, NN Control,* $\epsilon = 10^{-6}$

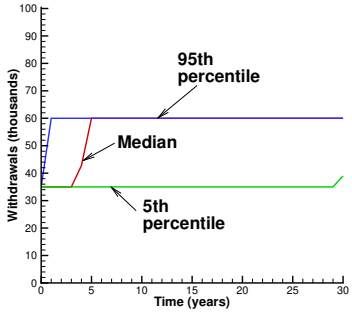
HJB Control Results (Positive and Negative Stabilization)



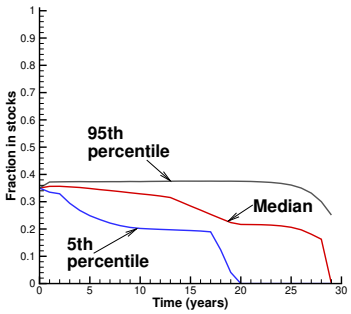
(d) *Percentiles fraction in stocks, HJB Control,* $\epsilon = 10^{-6}$



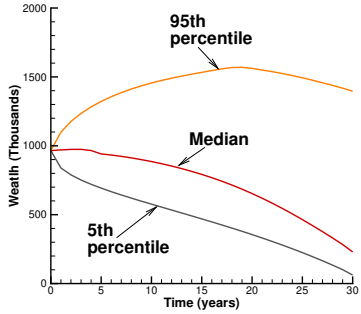
(e) *Percentiles wealth, HJB Control,* $\epsilon = 10^{-6}$



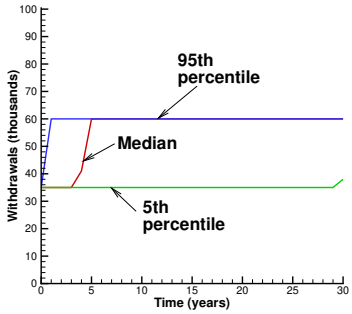
(f) *Percentiles withdrawals, HJB Control,* $\epsilon = 10^{-6}$



(g) *Percentiles fraction in stocks, HJB Control,* $\epsilon = -10^{-6}$



(h) *Percentiles wealth, HJB Control,* $\epsilon = -10^{-6}$



(i) *Percentiles withdrawals, HJB Control,* $\epsilon = -10^{-6}$

FIGURE 4.2.5: *Scenario in Table 4.2.1. NN and HJB controls computed from the problem (3.1.22). Parameters based on the real CRSP index, and real 10-year treasuries (see Table 4.1.1). NN model trained on* 2.56×10^5 *observations of synthetic data. HJB framework results from* 2.56×10^6 *observations of synthetic data.* $q_{min} = 35, q_{\max} = 60$, $\kappa = 1.0$. $W^* = 59.1$ *for NN results.* $W^* = 58.0$ *for HJB results. Units: thousands of dollars.*

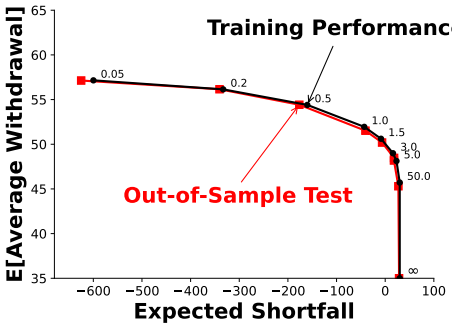


FIGURE 4.3.1: *Out-of-sample test. EW-ES frontiers, computed from the problem (3.1.22). Note: Scenario in Table 4.2.1. Comparison of NN training performance results vs. out-of-sample test. Both training and testing data are* 2.56×10^5 *observations of synthetic data, generated with a different random seed. Parameters for synthetic data based on cap-weighted real CRSP, real 10 year treasuries (see Table 4.1.1).* $q_{min} = 35, q_{\max} = 60$. $\epsilon = 10^{-6}$. *Units: thousands of dollars. Labels on nodes indicate* κ *parameter values.*

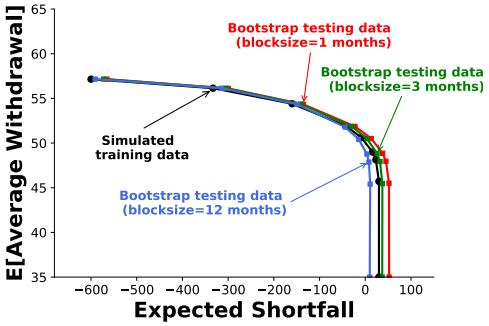


FIGURE 4.3.2: *Out-of-distribution test. EW-ES frontiers of controls generated by NN model trained on* 2.56×10^5 *observations of synthetic data, tested on* 2.56×10^5 *observations of historical data with varying expected block sizes. Computed from the problem (3.1.22). Note: Setup as in Table 4.2.1. Parameters based on real CRSP index and real 10-year U.S. Treasuries (see Table 4.1.1). Historical data in range 1926:1-2019:12. Units: thousands of dollars.* $q_{min} = 35; q_{max} = 60$. *Simulated training data refers to Monte Carlo simulations using the SDEs* (3.1.3) *and* (3.1.4).

Chapter 5

Dynamic Factor Investing

The NN framework developed in Section 3.4 avoids dynamic programming and value iteration, exploiting the low-dimensionality of the problem's control by solving for it directly. This allows us to extend the framework to a number of assets higher than 2 without significantly increasing the computational complexity. In this chapter, we detail this extended framework and use it to incorporate factor-based assets in the decumulation problem. In our computational experiments at the end of this chapter, we find promising evidence for the usefulness of factor investing for significantly improving risk management and withdrawal efficiency in the decumulation problem.

Factor investing has been one of the fastest growing areas in asset management for over a decade, with U.S.-listed equity factor ETFs having gone from less than $50 billion assets under management in 2009 to almost $500 billion in 2019 (Ang, 2019). In recent years, factor investing has also emerged as a commonly adopted response to an increasingly uncertain market environment. The *2022 Invesco Global Factor Investing Study* surveyed a group of asset managers with a collective $25.4 trillion assets under management. 67% of survey respondents said that factor investing had helped them manage market volatility in the past 12 months, and 41% said they had increased their factor asset allocations in that time (compared with 1% who decreased it) (Haghbin and Masse, 2022).

In this investigation, we aim to utilize factors that are widely accepted to have explanatory power over the cross-section of asset returns to determine if the inclusion of factor-based assets in our NN framework can improve investment performance in our decumulation problem. Another way to see this investigation's contribution is that it employs the NN framework to learn the historical distribution of factor assets and provide an optimal factor-weighting strategy accordingly. It also exploits the problem's long-term invest-

ment horizon to overcome the cyclicality of factor performance and provides the investor with more options in seeking a more favorable risk-reward trade-off in creating a portfolio decumulation strategy. This approach to factor investing could be seen as an alternative to strategies that employ *factor timing*. While there is some evidence for the predictability of factor performance (Haddad et al., 2020), much of the literature agrees that factor timing remains difficult and ill-advised in practice (Bender et al., 2018; Dichtl et al., 2019).

This investigation builds upon the work of Van Staden et al. (2022), which adapts the NN methodology proposed in Li and Forsyth (2019) to the dynamic factor investing problem, without decumulation. This chapter makes the following additional contributions:

(i) The novel NN approach for portfolio decumulation developed in Chapter 3 of this thesis is extended to include factor assets, and for a longer investment horizon than in Van Staden et al. (2022). A primary goal of this investigation is to determine how to optimally include factor assets to provide retirees with increased withdrawal efficiency and improved risk management.

(ii) This investigation also includes the implementation of additional constraints on the factor asset allocations in order to (1) better match practice, since asset managers would rarely advise a client to invest 100% of their portfolio into a factor asset; and (2) induce the NN to choose a more diversified strategy. As Van Staden et al. (2022) and this work find, allowing the NN to allocate 100% into a single asset leads it to choose only a few assets even if many are available. This investigation aims to determine if these additional constraints and the resulting increased asset diversification yield more robust out-of-distribution test results.

5.1 Formulation

In this section, we describe the updated formulation for the dynamic factor investing problem with decumulation (DFDC) to allow for the inclusion of additional candidate assets. We also introduce new optional constraints for the asset allocation control. Much of the original problem formulation described in Section 3.1 remains the same and we cover the differences here. Instead of only a stock index and a bond index, we now generalize to the case where the investor has access to a pre-determined set of $N_a \in \mathbb{N}$ investable assets which they may include in their portfolio. These assets can be distinguished into two categories: (1) "basic" assets, which include the same 10-Year Treasury and broad U.S. stock market index from the two-asset problem, as well as the U.S. 30-Day Treasury Bill

as a "risk-free" asset; and (2) "factor" assets, which are portfolios constructed by sorting all stocks by a chosen factor characteristic and selecting all stocks within a set percentile (e.g., the stocks within the 30th smallest percentile of market capitalization for the "Size" factor).

5.1.1 Notation

In general, let $A_i(t)$ specify the amount invested in asset $i \in 1, \ldots, N_a$ at time $t \in [0,T]$. At the beginning of the investment period, $W(t_0^-) = w_0$ and thereafter the total portfolio wealth is given by

$$W(t) = \sum_{i=1}^{N_a} A_i, \ t \in [0,T] \ . \tag{5.1.1}$$

We denote this multi-asset state space by $\tilde{X}(t) = A(t), t \in [0,T]$, where $A(t)$ is the vector of amounts invested in each asset $A_i, i \in \{0, \ldots, N_a\}$. In general the state space $\tilde{X}(t)$ can include additional variables that describe market conditions. As before, in computational practice, we only use total wealth, calculated from $A(t)$, and time as state variables for the control. In this case, we refer to the realized state as $x = \mathbf{a}$.

The allocation control for an asset $i \in 1, \ldots, N_a$ at time t_n is modeled as the proportion of total portfolio wealth to be invested in that asset and is denoted by $p_i(\tilde{X}_n^+, t_n)$:

$$p_i(\tilde{X}_n^+, t_n) = \frac{A_i(t_n^+)}{W(t_n^+)}, \ i \in \{1, \ldots, N_a\}, \ t_n \in \mathcal{T} \tag{5.1.2}$$

We can represent the allocation control for all assets at t_n with the vector $p(t_n) = p(\tilde{X}_n^+, t_n)$:

$$p(\tilde{X}_n^+, t_n) = [p_1(t_n), \ldots, p_{N_a}(t_n)], \ t_n \in \mathcal{T} \tag{5.1.3}$$

The withdrawal control at time t_n, $q(t_n) = q(\tilde{X}(t_n^-), t_n)$, is modeled and constrained the same way as in the previous formulation. We will also continue the convention of considering both the withdrawal and allocation control as functions of wealth before and after withdrawal:

$$q(t_n) = q(\tilde{X}(t_n^-), t_n) = q(W(t_n^-), t_n)$$
$$p(t_n) = p(\tilde{X}(t_n^+), t_n) = p(W(t_n^+), t_n) \tag{5.1.4}$$

We continue to assume instantaneous rebalancing, meaning there are no changes in asset prices in the interval (t_n^-, t_n^+). For the decumulation problem with more than 3 assets, a control at time t_n is described by a pair $(q(t_n), p(t_n)) \in \tilde{\mathcal{Z}}(W_n^-, W_n^+, t_n)$, where $\tilde{\mathcal{Z}}$ represents the set of admissible controls. The constraints of no shorting and no leverage are imposed on the allocation control. In the case of insolvency, trading will cease and debt will accumulate at the borrowing rate (approximated by the 10-Year U.S. Treasury rate). These constraints are true for all portfolios and are the same as in the previous formulation. However, some portfolio constructions in this factor investing investigation will impose new constraints on the maximum allocation proportions for factor-based assets. The portfolios considered in this section will either:

(i) have no additional limits on asset allocations besides no leverage and no shorting ($limit\_type = none$),

(ii) individually limit each factor-based asset to a maximum proportion of total wealth ($limit\_type = indiv.$); or,

(iii) limit the maximum proportion that can be allocated to the factor-based assets as a group ($limit\_type = group$).

Let N_f be the number of candidate assets that are factor-based assets, a subset of A. It follows that $N_f < N_a$. The factor-based assets will be indexed by $i \in \{1, \ldots, N_f\}$. Then we define the subset of representing the non-factor assets (basic assets) to be indexed by $i \in \{N_f + 1, \ldots, N_a\}$.

All portfolios' candidate basic assets will include 10-Year U.S. Treasury Notes (B10) and for notational simplicity we will always have it represented by index $i = N_a$. We can therefore represent the allocation control in the case of insolvency as the vector $(0, \ldots, 0, 1)$, indicating that debt is to accumulate at the rate determined by the B10 asset. As in the previous problem formulation, we assume that the investor can obtain a reverse mortgage on real estate in order to cover their investment shortfall. We use B10 to approximate the U.S. fixed mortgage rate since the mortgage rate has historically closely tracked the 10-Year Treasury yield, although the spread has been at least 2% in recent years (Edelberg and Steinmetz-Silber, 2023).

At the end of the investment horizon, $t = T$, the portfolio is liquidated and all wealth is allocated to the 30-Day Treasury Bill (T30) as a cash equivalent. The T30 asset will be indexed as $i = N_a - 1$. We can therefore represent the allocation control at $t = T$ as the vector $(0,\ldots,1,0)$, indicating that 100% of wealth is to be allocated to the T30 asset.

The admissible control for the dynamic factor investing problem with decumulation can be represented mathematically as the following:

$$\tilde{\mathcal{Z}}_p(W_n^+, t_n) = \begin{cases} (y_1,\ldots,y_{N_a}) \in \mathbb{R}^{N_a} \ : \ \sum_{i=1}^{N_a} y_i = 1, y_i \geq 0 & \forall i \in \{1,\ldots,N_a\}; \\ & W_n^+ > 0; \ t_n \in \mathcal{T} \ ; \ t_n \neq t_M \\ y_i \leq max_{indiv} \ \forall i \in \{1,\ldots,N_f\} & \text{if } limit\_type = indiv. \\ \sum_{i=1}^{N_f} y_i \leq max_{group} & \text{if } limit\_type = group \\ (0,\ldots,0,1) \in \mathbb{R}^{N_a}; & W_n^+ \leq 0; \ t_n \in \mathcal{T} \ ; \ t_n \neq t_M \\ (0,\ldots,0,1,0) \in \mathbb{R}^{N_a}; & t_n = T \end{cases} \tag{5.1.5}$$

Where $max_{group} < 1$ and $max_{indiv} = 1/N_f \cdot max_{group}$. Therefore, the maximum proportion allocated to all factor assets will be max_{group}.

The admissible withdrawal set, $\mathcal{Z}_q(W_n^-, t_n)$, is unchanged from the previous formulation in (3.1.10).

$$\tilde{\mathcal{Z}}(W_n^-, W_n^+, t_n) = \mathcal{Z}_q(W_n^-, t_n) \times \tilde{\mathcal{Z}}_p(W_n^+, t_n) \tag{5.1.6}$$

At each rebalancing time, we seek the feasible optimal control for all possible combinations of $A_i(t) \ : \ i \in \{1,\ldots,N_a\}$, that result in the same total wealth. As before, we will refer to the allocation control as a function of wealth and time after withdrawal, (W_n^+, t_n). The admissible control set for the dynamic factor investing problem $\tilde{\mathcal{A}}$ can be written as

$$\tilde{\mathcal{A}} = \left\{ (q(t_n), p(t_n))_{0 \leq n \leq M} : (q(t_n)), p(t_n)) \in \tilde{\mathcal{Z}}(W_n^-, W_n^+, t_n) \right\}. \tag{5.1.7}$$

An admissible control for the dynamic factor investing problem $\tilde{\mathcal{P}} \in \tilde{\mathcal{A}}$, can be written as

$$\tilde{\mathcal{P}} = \{(q(t_n), p(t_n)) \ : \ n = 0, \ldots, M\} \ . \tag{5.1.8}$$

5.1.2 Asset and Wealth Dynamics

Given a control $\mathcal{P}$, each rebalancing time consists of the following events:

- The investor observes the amounts in each portfolio asset. These amounts, $A_i(t_n^-)$, are calculated from the amounts after rebalancing in the previous rebalancing step, $A_i(t_{n-1}^+)$, together with the observed return $R_i(t_n)$ from the interval $[t_{n-1}^+, t_n^-]$:

$$A_i(t_n^-) = A_i(t_{n-1}^+) \cdot [1 + R_i(t_{n-1})], \quad i = 1, \ldots, N_a \tag{5.1.9}$$

- The investor determines an amount, q_n to withdraw according to the control $q(\cdot)$. The control is a function of the current wealth, which is obtained according to (5.1.1).

$$W(t_n^+) = W(t_n^-) - q_n \tag{5.1.10}$$

- The investor determines the new amounts invested in each asset $i \in \{1, \ldots, N_a\}$ based on $W(t_n^+)$ and according to the control p.

$$A_i(t_n^+) = W(t_n^+) \cdot p_i(t_n), \qquad i = 1, \ldots, N_a. \tag{5.1.11}$$

A formulation including transaction costs is outside the scope of this work. Note that in Dang and Forsyth (2014), it is shown that the transaction costs typical of liquid ETFs have an insignificant effect on the results.

5.2 Neural Network Model for Factor Investing with Decumulation

We now develop the neural network model to solve the formulation described in Section 5.1. This NN model builds upon the one proposed in Section 3.4 to simultaneously solve for the allocation and withdrawal controls. In this section, we review the NN methodology and detail how it is extended to include multiple assets and optional additional constraint functions.

As before in Section 3.4, we consider two fully-connected, feed-forward neural networks. We approximate the control in $\tilde{\mathcal{P}}$ directly by using feed-forward, fully-connected neural networks. Given parameters $\boldsymbol{\theta}_p$ and $\boldsymbol{\theta}_q$, i.e. NN weights and biases, $\hat{p}(W(t_n), t_n, \boldsymbol{\theta}_p)$ and $\hat{q}(W(t_n), t_n, \boldsymbol{\theta}_q)$ approximate the controls p_n and q_n respectively,

$$\begin{gathered}\hat{q}(W_n^-, t_n^-, \boldsymbol{\theta}_q) \simeq q(W_n^-, t_n) \ ; \ n = 0, \ldots, M \\ \hat{p}(W_n^+, t_n^+, \boldsymbol{\theta}_p) \simeq p(W_n^+, t_n) \ ; \ n = 0, \ldots, M-1 \\ \hat{\mathcal{P}} = \{(\hat{q}(\cdot), \hat{p}(\cdot))\} \simeq \tilde{\mathcal{P}}\end{gathered}$$

As before, time and wealth are the only inputs to the NNs. We will now proceed to detail how the structure of the NN $\hat{p}$ is adapted to the factor investing problem, including the optional new constraint functions. The structure of the NN $\hat{q}$ is unchanged.

5.2.1 Objective Function

In order to be able to effectively compare the results of the dynamic factor strategies developed in this chapter to the results of the two-asset problem, we will continue to use the mean-CVaR objective function. However, since the market model and constraints have changed, we state the new formulation of the objective here.

Our goal is to choose NN weights $\boldsymbol{\theta}_p$ and $\boldsymbol{\theta}_q$ by solving (3.1.22), with $\hat{q}(W_n^-, t_n^-, \boldsymbol{\theta}_q)$ and $\hat{p}(W_n^+, t_n^+, \boldsymbol{\theta}_p)$ approximating feasible controls $(q(\cdot, t_n), p(\cdot, t_n)) \in \tilde{\mathcal{Z}}(W_n^-, W_n^+, t_n)$ for $t_n \in \mathcal{T}$. For an arbitrary set of controls $\hat{\mathcal{P}}$ and wealth level W^*, we define the NN performance criteria for the DFDC problem $\tilde{V}_{NN}$ as

$$\begin{aligned}&\tilde{V}_{NN}(\boldsymbol{\theta}_q, \boldsymbol{\theta}_p, W^*, \mathbf{a}, t_0^-) = \\ &\tfrac{1}{N}\sum_{j=1}^{N}\left[\sum_{n=0}^{M}\hat{q}((W_n)^j, t_n; \boldsymbol{\theta}_q) \ + \ \kappa\left(W^* + \frac{1}{\alpha}\min((W_T)^j - W^*, 0)\right) + \epsilon(W_T)^j \,\middle|\, X_0^- = \mathbf{a}\right] \\ &\text{subject to} \begin{cases} (A(t_n))^j \text{ calculated using the } j^{th} \text{ sample of returns, according to (5.1.9)}; \quad t \notin \mathcal{T} \\ (W_n^+)^j = (\sum_{i=1}^{N_a} A_i)^j - \hat{q}\left((W_{t_n}^-)^j, t_n, \boldsymbol{\theta_q}\right) \ ; \ (X_n^+)^j = (A(t_n^+))^j \\ (A(t_n)^+)^j = \hat{p}\left((W_n^+)^j, t_n, \boldsymbol{\theta_p}\right) \ (W_n^+)^j \\ (\hat{q}(\cdot), \hat{p}(\cdot)) \in \tilde{\mathcal{Z}}\left((W_n^-)^j, (W_n^+)^j, t_n\right) \\ n = 0, \ldots, M \ ; \ t_n \in \mathcal{T} \end{cases},\end{aligned} \tag{5.2.1}$$

5.2.2 Activation Functions for Asset Allocation Constraints

In this section, we describe the custom activation functions that encode the constraints $\tilde{\mathcal{Z}}$ defined in (5.1.5).

The output layer of the multi-asset allocation NN is defined to have η nodes. $\eta = N_a$ unless $limit\_group = True$, in which case $\eta = N_a + 1$. The value of the i-th output node representing the NN approximation of the optimal control for asset $i \in \{1, \ldots, N_a\}$, $p_i(\cdot)$. When included in the NN structure, $(N_a + 1)$-th node is used to determine the total factor proportion in the case of $limit\_type = group$. The output vector of the NN can be expressed as

$$\hat{p}(\cdot) = \sigma_{[limit\_type]}(z), \tag{5.2.2}$$

where $z \in \mathbb{R}^{N_a}$ is the vector of values from each node of the output layer before the application of the activation function, $\sigma(\cdot)$.

In the two-asset problem the activation function $\sigma(\cdot)$ was simply the softmax function. In the following, we detail the new activation functions depending on the asset constraint imposed for the three possible choices of asset constraints. Each activation function will be a concatenation of the outputs of the logistic sigmoid and softmax functions applied to different nodes of the output layer. We define these component functions here and then enumerate the new allocation constraint functions considered in this investigation.

$$Sigmoid(z_i) = \frac{1}{1 + \exp(-z_i)}, \ j \in \{1, \ldots, N_a + 1\} \tag{5.2.3}$$

$$Softmax\big((z_1, \ldots, z_K)\big) = \begin{bmatrix} \exp(z_1) \\ \vdots \\ \exp(z_K) \end{bmatrix} \cdot \frac{1}{\sum_{j=1}^{K} \exp(z_j)} \tag{5.2.4}$$

(i) $limit\_type = none$, where all nodes are inputs to one instance of the softmax activation function (typical softmax);

$$\sigma_{[none]}(z) = Softmax\big((z_1, \ldots, z_{N_a})\big) \tag{5.2.5}$$

The vector output of the softmax function is guaranteed to be non-negative and sum to one, ensuring that the output of the NN satisfies the constraint (5.1.5).

(ii) $limit\_type = indiv.$, in which the activation is a concatenation of a softmax function and logistic sigmoid functions. Nodes representing the control for basic assets $i \in \{N_f+1, \ldots, N_a\}$ are the inputs to one instance of the softmax function while a sigmoid function is applied to each factor asset node $i \in \{1, \ldots, N_f\}$. The outputs of each of these components are concatenated to form the output of the $\mathbb{R}^{N_a}$-valued function.

$$\sigma_{[indiv.]}(z) = \begin{bmatrix} Sigmoid(z_1) \cdot max_{indiv} = a_1 \\ \vdots \\ Sigmoid(z_{N_f}) \cdot max_{indiv} = a_{N_f} \\ \\ Softmax\big((z_{N_f+1}, \ldots, z_{N_a})\big) \cdot (1 - \sum_{i=1}^{N_f} a_i) \end{bmatrix}, \tag{5.2.6}$$

where max_{indiv} indicates the predetermined maximum proportion to be allocated to each factor asset $i \in \{1, \ldots, N_f\}$. Components $a_i, \ldots, a_{N_f}$ are calculated first so that the softmax outputs can be scaled appropriately. The sum of all components of (5.2.6) are be guaranteed to be 1:

$$\begin{aligned} \sum_{i=1}^{N_a} \big[\sigma_{[indiv.]}(z)\big]_i = \sum_{i=1}^{N_f} \big[Sigmoid(z_i)\big] \cdot max_{indiv} + \\ \sum_{i=N_f+1}^{N_a} \big[Softmax\big((z_{N_f+1}, \ldots, z_{N_a})\big)\big]_i \cdot (1 - \sum_{i=1}^{N_f} \big[Sigmoid(z_i)\big] \cdot max_{indiv}) = \\ \sum_{i=1}^{N_f} \big[Sigmoid(z_i)\big] \cdot max_{indiv} + 1 - \sum_{i=1}^{N_f} \big[Sigmoid(z_i)\big] \cdot max_{indiv} = 1 \end{aligned} \tag{5.2.7}$$

(iii) $limit\_type = group$, in which the activation is a concatenation of 2 softmax functions, one applied to nodes representing factor asset controls and one applied to nodes representing basic asset controls. The total proportion allocated to the group of factor assets is determined by node $N_a + 1$, with sigmoid activation applied.

$$\sigma_{[group]}(z) = \begin{bmatrix} Softmax\big((z_1,\ldots,z_{N_f})\big) \cdot Sigmoid(z_{N_a+1}) \cdot max_{group} \\ \\ Softmax\big((z_{N_f+1},\ldots,z_{N_a})\big) \cdot \big(1 - Sigmoid(z_{N_a+1}) \cdot max_{group}\big) \end{bmatrix} \tag{5.2.8}$$

The components of this function's output are guaranteed to sum to one:

$$\sum_{i=1}^{N_a} \big[\sigma_{[group]}(z)\big]_i = \sum_{i=1}^{N_f} \big[Softmax\big((z_1,\ldots,z_{N_f})\big)\big]_i \cdot Sigmoid(z_{N_a+1}) \cdot max_{group} +$$
$$\sum_{i=N_f+1}^{N_a} \big[Softmax\big((z_{N_f+1},\ldots,z_{N_a})\big)\big]_i \cdot \big(1 - Sigmoid(z_{N_a+1}) \cdot max_{group}\big) =$$
$$Sigmoid(z_{N_a+1}) \cdot max_{group} + \big(1 - Sigmoid(z_{N_a+1}) \cdot max_{group}\big) = 1$$

5.2.3 Neural Network Structure

One of the key features of the NN framework developed in Section 3.4 is its scalability as problem complexity increases. In order to adapt to the multi-asset problem as described above, we make only the following adjustments to the NN architecture:

(i) Increase the number of nodes in the output layer to N_a, or $N_a + 1$ in the case of $limit\_type = group$.

(ii) Change the activation function of the allocation network output layer to (5.2.6) if $limit\_type = indiv.$, or to (5.2.8). if $limit\_type = group$. If $limit\_type = none$, the activation is the original softmax function as in the previous formulation.

(iii) Increase the number of nodes in each hidden layer to $N_a + 8$.

As will be shown in Section 5.5, these remarkably modest adjustments to the NN structure will allow it to effectively compute decumulation and allocation strategies for a greatly expanded problem space. The updated NN structure is illustrated in Figure 5.2.1.

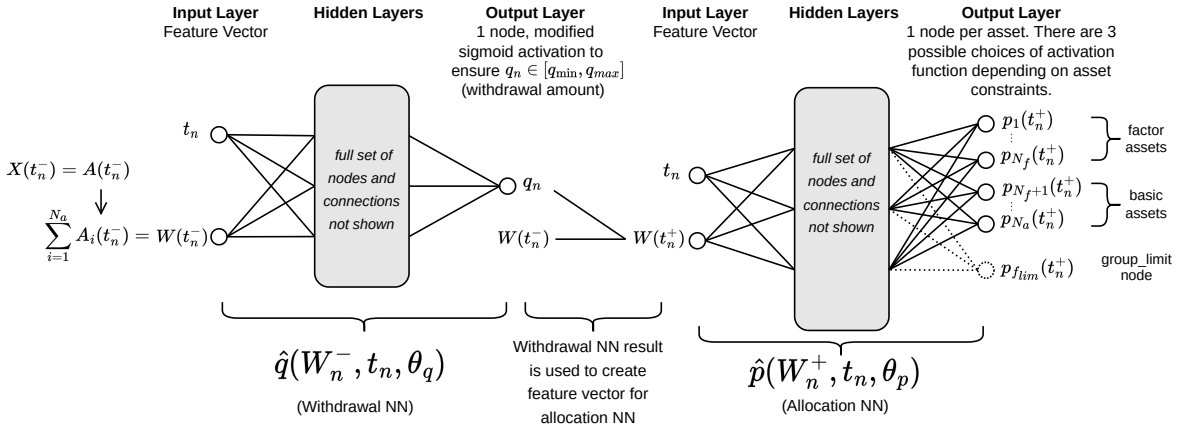


FIGURE 5.2.1: *Illustration of the NN framework as per Section 5.2. Additional technical details can be found in 3.4.2.*

5.3 Investment Scenario and Factor Selection

The investment scenario for the following computational experiments will be similar to that of Chapters 3 and 4, except that the investor will have access to several more assets and will include additional asset constraints. These assets will always be modeled using historical data, not synthetic data. All return data are inflation-adjusted. We describe these assets here.

The basic assets consist of (i) the 30-day U.S. Treasury Bill ("T30"), which can be considered the "risk-free" asset [1], (ii) a 10-Year U.S. Treasury Bond, and (iii) a capitalization-weighted stock index, which aligns closely to the definition of a "Market Portfolio" in factor models.

The selection of which factor-based assets to use as candidate assets is a critical decision for the investor. However, as discussed in Chapter 2, there is little consensus in the literature on which combination of factors has the most power to explain the cross-section of expected returns across assets (Harvey et al., 2015). To avoid this controversial issue, we will only consider those factors with the widest acceptance in both the academic literature and in practice by the asset management industry. These recognized factors that we consider are (i) Size, (ii) Value, (iii) Momentum, and (iv) Low Volatility ("Vol"). The recognition of these factors can be seen in their prevalence in the asset management industry: the five largest ETF issuers in the U.S. (BlackRock, Vanguard, State Street, Invesco,

[1] since the T-bill returns are inflation-adjusted, this asset is not truly risk-free in real terms.

and Charles Schwab) all offer ETF products based upon each of these factors. They are popular in the literature as well. The Size and Value factors are perhaps the most famous and classical factors, being the 2 factors complementing the Market factor in the seminal 3-Factor Model proposed by Fama and French (1993). The newer additions, Momentum and Low Volatility, have also gained mainstream acceptance, such as in Asness et al. (2013) and Dutt and Humphery-Jenner (2013). The specific definition of each of these factors is given alongside accompanying data definitions in Table 5.4.1.

We have already cited the proliferation of factor-based ETFs as evidence for our selected factors' widespread recognition. The popularity and accessibility of these ETFs further motivate our selection of these factors since we seek to develop dynamic factor investing strategies that are investable. Academic research typically focuses on factors formulated as a combination of a "long leg" and a "short leg", such as the Size factor which consists of a long position in small stocks and a short position in large stocks. This serves the academic purpose of better capturing the effect of stock size on the cross-section of returns while maintaining zero exposure to the market factor but is not meant to be an investable asset for most investors. Fortunately, there is considerable evidence that investing in just the long leg of a factor is enough to provide an investor with meaningful exposure to that factor (Blitz et al., 2020; Israel and Moskowitz, 2013). We therefore assume that the investor will choose only recognized factor ETFs as candidate assets, and base our experiments on this assumption.

Basket Label	Basic Assets			Factor Assets				N_a
	B10	Mkt	T30	Size	Value	Mom	Vol	
2 Basic	✓	✓						2
3 Basic	✓	✓	✓					3
2 Factor	✓	✓	✓	✓	✓			5
3 Factor	✓	✓	✓	✓	✓	✓		6
4 Factor	✓	✓	✓	✓	✓	✓	✓	7

TABLE 5.3.1: *Asset baskets considered in the analysis. The "✓ " indicates inclusion in the asset basket. Each factor portfolio considered will include one of the factor asset baskets listed here, as well as all 3 basic assets.*

5.4 Data

For basic assets, we continue to use CRSP data as in Section 4. Note that we will use different data ranges to match the availability of factor data.

Investment horizon T (years)	30
Market Assets	Described in Tables 4.1 and 5.3.1
Initial portfolio value W_0	1000
Cash withdrawal times	$t = 0,1,\ldots,30$
Withdrawal range	$[35, 60]$
Asset allocation constraints	Defined in (5.1.5)
max_{group}	50%
$max_{indiv.}$	max_{group}/N_f
Borrowing spread μ_c^b	0.0
Rebalancing interval (years)	1

TABLE 5.3.2: *Dynamic factor investing problem setup and input data. Monetary units: thousands of dollars.*

A factor-based asset is essentially a portfolio of stocks selected to gain exposure to a desired characteristic. The Kenneth French Data Library [2] (KFDL) provides historical data for an extensive range of portfolios constructed in this way. The data is openly accessible, transparent in its construction, and more extensive than data for any proprietary ETF. Therefore, we proceed with our experiments confident that the KFDL data provides a good proxy for investable factor-based assets. All factor asset returns used in the following experiments were adjusted for inflation using the CPI index from CRSP. Van Staden et al. (2022) finds that returns of popular factor ETF products have a high correlation with our chosen proxy data, generally exceeding 0.90. The data sets for each factor are defined in Table 5.4.1.

In order to generate training and testing data sets, we will use the same bootstrap resampling methodology described in 4.1 applied to this new data set. For the following experiments, we will generate 2 different pairs of training and testing data sets based on the CRSP and KFDL data. These data sets are defined in Table 5.4.2. In DS1, we use the same date range of data and construct an out-of-sample test data set by resampling with a different expected block size. See Ni et al. (2022a) for a proof of how the bootstrapping methodology effectively guarantees that the probability of an identical path in the training and testing data sets is vanishingly small. In DS2, we split the available data by date to create an out-of-distribution test set. Note that the Low Volatility factor, "Vol", only has data availability beginning July 1963, while the others have availability beginning in

[2]Kenneth R. French's Data Library can be accessed at: http://mba.tuck.dartmouth.edu/pages/faculty/ken.frenchinput/data_library.html.

Asset	Source and Description
Basic: T30	CRSP: U.S. 30-Day Treasury Bill, monthly real returns. Available 1927:01-2022:12.
Basic: B10	CRSP: U.S. 10-Year Treasury Note, monthly real returns. Available 1927:01-2022:12.
Basic: Mkt	CRSP: Capitalization-weighted domestic stock index, monthly real returns. Available 1927:01-2022:12.
Factor: Size	KFDL: "Portfolios formed on Size": Monthly real returns of capitalization weighted index containing U.S. stocks with market equity in lowest 30th percentile. Available 1927:07-2022:12.
Factor: Value	KFDL: "Portfolios formed on Book-to-Market": Monthly real returns of capitalization-weighted index containing U.S. stocks with book-to-market equity in highest 30th percentile. Available 1927:07-2022:12.
Factor: Mom	KFDL: "Portfolios formed on Prior Returns": Monthly real returns of equal-weighted average of two capitalization-weighted indices of U.S. stocks returns which consist of firms below or above the median market capitalization. Each subindex includes only those firms that have returns from the prior 2-12 month period in the highest 30th percentile. Available 1927:07-2022:12.
Factor: Vol	KFDL: "Portfolios formed on Variance": Monthly real returns of capitalization-weighted index containing U.S. stocks with daily return variance during preceding 60 days below 20th percentile. Available 1963:07-2022:12.

TABLE 5.4.1: *Sources and definitions of data used for candidate assets used in this analysis. All time series are inflation-adjusted by using U.S. CPI data from CRSP.*

July 1927. In order to construct a large enough training data set for DS2, we omit the "4 Factor" asset basket from experiments with the DS2 data set.

	Training	Testing
DS1	Date range: 1963:07-2022:12, Exp. block size = 6 Months	Date range: 1963:07-2022:12, Exp. block size = 18 Months
DS2	Date range: 1927:07-1991:12, Exp. block size = 6 Months	Date range: 1992:01-2022:12, Exp. block size = 6 Months

TABLE 5.4.2: *Data set combinations used for training and testing in computational experiments. Size of all data sets is* $N_d = 2.56 * 10^6$.

Figures 5.4.1, 5.4.3, and 5.4.2 provide summary statistics for the chosen candidate assets' returns and their historical performance.

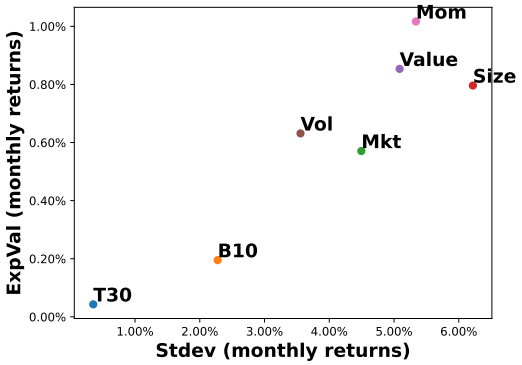


FIGURE 5.4.1: *Expected value and standard deviation of monthly returns of each candidate asset, 1963:07-2022:12. Data is described in Table 5.4.1.*

Corr.	T30	B10	Mkt	Size	Value	Vol	Mom
T30	1.00						
B10	0.35	1.00					
Mkt	0.06	0.09	1.00				
Size	0.01	0.01	0.87	1.00			
Value	0.01	0.03	0.91	0.91	1.00		
Vol	0.08	0.20	0.53	0.36	0.42	1.00	
Mom	0.05	0.07	0.94	0.90	0.87	0.48	1.00

TABLE 5.4.3: *Correlation matrix of monthly real returns, 1963:07-2022:12. Data is described in Table 5.4.1.*

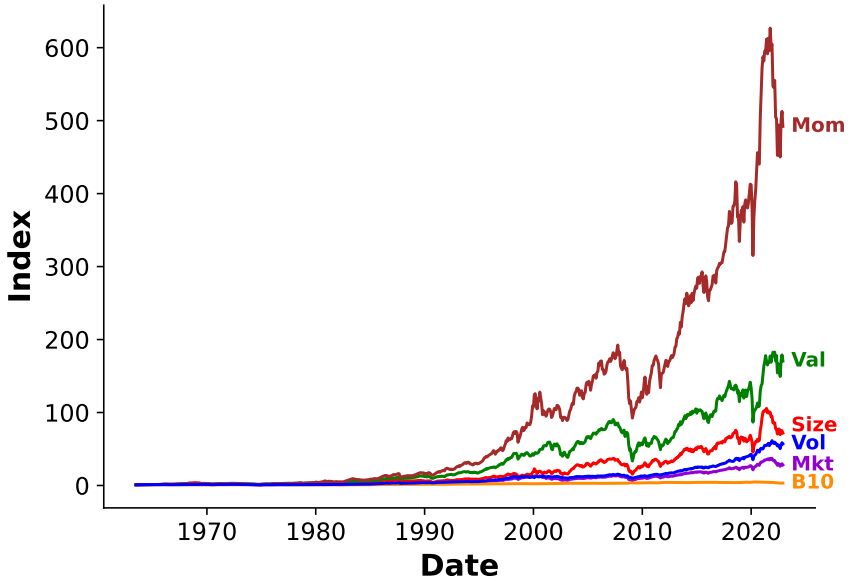


FIGURE 5.4.2: *Cumulative real return indexes of all candidate assets, 1963:07-2022:12. Data is described in Table 5.4.1.*

5.5 Computational Results

In this section, we discuss the computational results of applying the extended NN methodology presented in Section 5.2 on the dynamic factor investing problem as formulated in Section 5.1. The investment scenario is described in Table 5.3.2. The NN framework extended for this factor investing problem yielded the following results and appeared to converge when using the same hyperparameters as described in Table 3.5.1, a notable result since the NN structure and problem space have both become larger. Note that we only compare the investment performance of each NN strategy type with one risk-reward weighting of $\kappa = 1.0$. We will trace the efficient frontier for the "2 Factor" (Size and Value) asset basket with $limit\_type = indiv.$ for comparison with the basic asset NN strategy. Also, note that the results for the "2 Basic" NN strategy may appear to be slightly different from the results in Chapter 4; this is due to the different date ranges of data used in the following experiments.

5.5.1 Constant Benchmarks

Each asset basket in Table 5.3.1 is also used to create a constant proportion strategy to serve as a benchmark for comparison with each of the NN strategies applied to these asset baskets.

At each rebalancing time t_n, the constant strategy is executed as follows:

(i) Withdraw 40 wealth.

(ii) Allocate 50% of wealth to asset B10. Allocate $\frac{1}{2(N_f+1)}$ of wealth to each factor asset and "Mkt" asset. The portfolio is essentially 50% B10 and 50% an equal distribution across equity-based assets.

5.5.2 Results: DS1

We first discuss the results for data set DS1, where both training and testing data sets are based on the data from July 1963 to December 2022, with different expected block sizes.

Table 5.5.1 shows the training and testing performance of each NN strategy and the associated benchmarks. We observe that every NN strategy with a "Factor" asset basket has better training performance compared to either "Basic" asset strategy. Indeed, upon

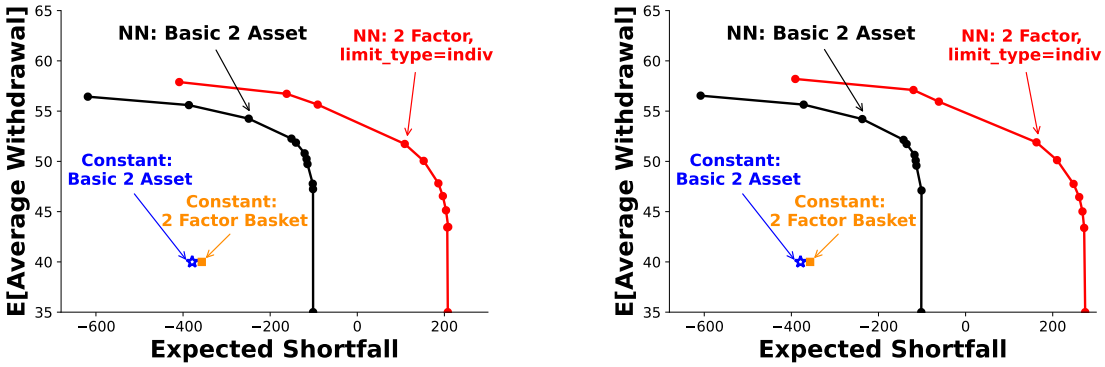
inspecting the resulting controls from each optimal NN strategy we find that all optimal NN strategies for "Factor" asset baskets do include significant allocations in factor assets in large regions of the time-wealth space. Beyond this commonality, there is significant variation among the NN strategies in the details of the asset allocation.

For all NN strategies with $limit\_type = none$ or $limit\_type = group$, the basic assets "T30" and "B10" were invested in some regions of the time-wealth space, while the "Mkt" asset never was. The Value factor always received investment in some regions, and the Momentum factor received the most investment when available. In the "4 Factor" asset basket, the Low Volatility factor notably did not receive any investment, while Value and Momentum continued to have non-zero weights. When given the freedom to do so, the NN strategy will determine that a fairly un-diversified strategy is optimal by forgoing any investment in the "Mkt" asset or the Size and Low Volatility factors. This is consistent with the findings of Van Staden et al. (2022).

Referencing Figure 5.4.1, we can see that the Value and Momentum factors had the highest expected monthly return of candidate assets over the DS1 time period. The outperformance of these two factors is also consistent with the findings of Asness et al. (2013), which was able to establish the consistent presence of return premia for the two factors in several different markets. Asness et al. (2013) also finds that the two factors (academic long-short versions) are negatively correlated, perhaps contributing to the optimality of a strategy that combines allocations in the two factors. Figure 5.4.2 shows the index of cumulative returns over the same period, again clearly showing the outperformance of Value and Momentum over the entire period, albeit with more volatility than other assets.

We now focus on analyzing the control for a single NN dynamic factor investing strategy. We have selected the "2 Factor" asset basket (Size, Value, and basic assets) with $limit\_type = indiv$ as an illustrative example of the optimal NN strategies for dynamic factor investing. Figure 5.5.1 compares the efficient frontiers of EW-ES performance for our selected factor strategy and the original basic 2 asset strategy. The efficient frontiers for the factor strategy are significantly to the right of the basic asset strategy's efficient frontiers, for both training and testing performance. This suggests that the NN framework is able to utilize the factor assets to create an investment strategy with a more favorable risk-reward trade-off compared with the basic assets alone. This result also holds in an out-of-sample test. Also interesting to note is that the constant proportion benchmark has very similar performance as the basic asset constant benchmark, indicating that the dynamic strategy created by the NN is significant in making effective use of the factor assets.

We proceed by interpreting the strategy given by the NN for the "2 Factor" asset



(a) *DS1, Training*

(b) *DS1, Testing*

FIGURE 5.5.1: *EW-ES frontiers of controls generated by NN model for the basic asset basket and the 2-factor asset basket (Size, Value, and basic assets) with limit_type = indiv. Computational results for DS1, defined in Table 5.4.2. NN methodology defined in Section 5.2. Investment scenario presented in Table 5.3.2. Points showing the performance of associated constant strategy benchmarks are also included. The constant strategy with "Basic 2 Asset" portfolio is essentially the Bengen 4% strategy with fixed 4% withdrawals and constant allocation of 30% in stocks, which was found to be the best-performing constant allocation. The constant benchmark strategy for the 2-factor basket is described in subsection 5.5.1.*

Results: DS1

NN Strategy		Training			Testing		
Asset Basket	*limit_type*	ES (5%)	Avg. q_i	Obj. Fun.	ES (5%)	Avg. q_i	Obj. Fun.
2 Basic	none	-148.3	52.2	1469.3	-142.3	52.1	1472.4
3 Basic	none	-106.4	50.9	1471.0	-100.4	50.8	1473.1
2 Factor	none	307.5	51.9	1917.2	436.5	52.1	2051.7
	group	126.2	53.0	1770.2	199.1	53.2	1849.4
	indiv.	109.8	51.7	1713.4	163.5	51.9	1772.5
3 Factor	none	608.8	53.0	2250.8	723.3	53.6	2384.4
	group	368.8	52.4	1993.7	429.0	52.6	2059.7
	indiv.	249.4	51.5	1845.8	294.6	51.8	1899.1
4 Factor	none	609.4	52.9	2250.2	746.6	52.7	2393.0
	group	364.7	52.4	1989.4	437.2	52.4	2062.7
	indiv.	154.7	51.9	1763.6	185.2	51.6	1783.7

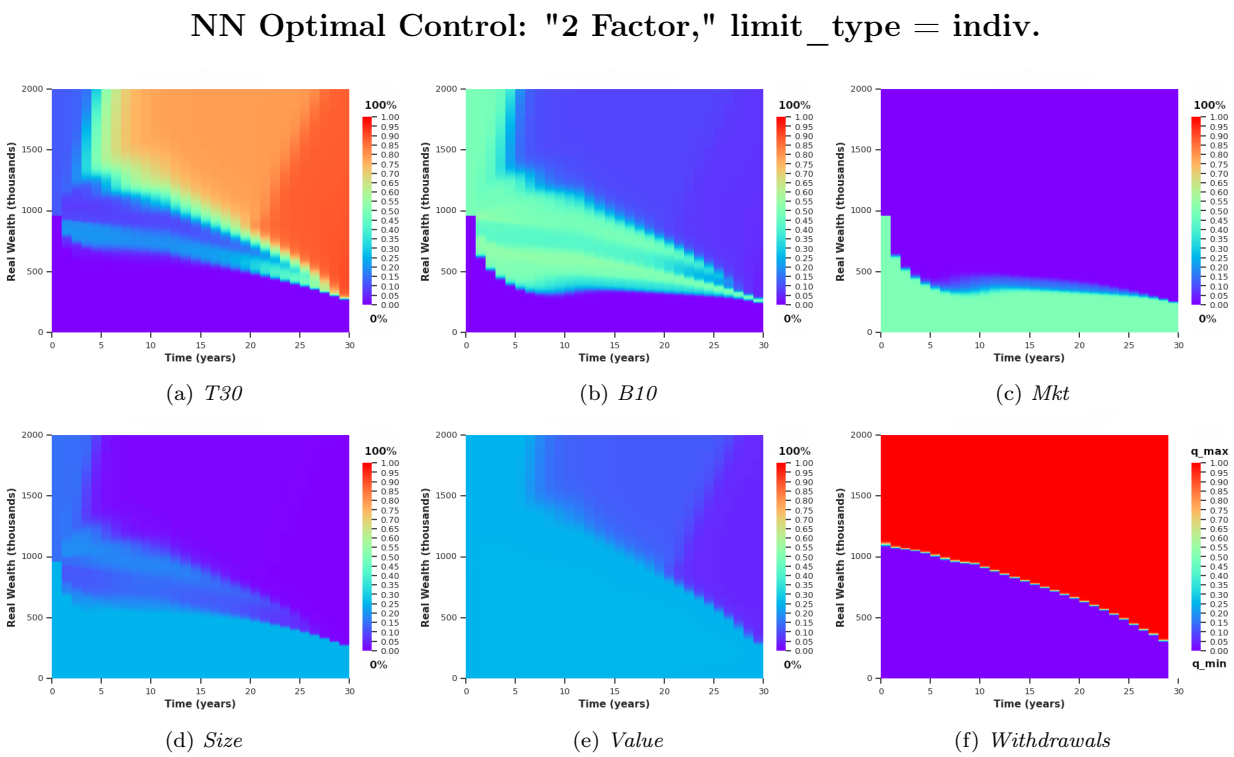
Constant Benchmark	Training			Testing		
Asset Basket	ES (5%)	Avg. q_i	Obj. Fun.	ES (5%)	Avg. q_i	Obj. Fun.
2 Basic	-375.9	40	864.1	-400.6	40	839.4
3 Basic	-395.3	40	844.7	-424.4	40	815.6
2 Factor	-354.9	40	885.1	-299.5	40	940.5
3 Factor	-301.6	40	938.4	-280.3	40	959.7
4 Factor	-300.4	40	939.6	-277.4	40	962.6

TABLE 5.5.1: *Computational results for DS1, defined in Table 5.4.2. NN methodology defined in Section 5.2. Investment scenario in Table 5.3.2. Constant benchmark strategies described in 5.5.1.* $\kappa = 1.0$.

basket with $limit\_type = indiv$. We will also describe the key differences between this control with the NN strategies with different asset baskets and asset constraints. Heat maps representing the control for our example strategy are presented in Figure 5.5.2 and percentile plots in Figure 5.5.3.

The $limit\_type = indiv$ allocation constraint had the expected result of inducing the NN strategy to invest in every available asset in some region of the time-wealth space. For an illustrative example of these multi-asset controls generated by the NN model, we represent them as heat maps in Figure 5.5.2. As $limit\_type = indiv$ is the most restrictive type of asset constraint tested in these experiments, it follows that the strategies with this limit had lower training performance compared to those with the more lenient constraints.

The withdrawal control, shown in heat map 5.5.2(f) and percentile plot 5.5.3(c), is



(a) *T30* (b) *B10* (c) *Mkt*

(d) *Size* (e) *Value* (f) *Withdrawals*

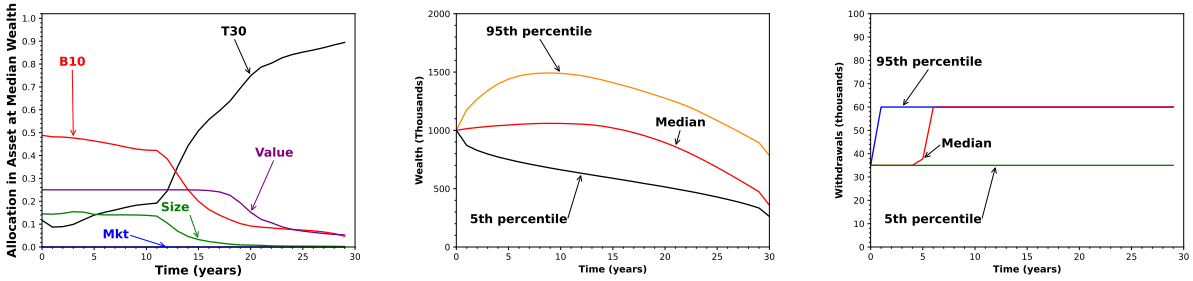
FIGURE 5.5.2: *Optimal control computed for "2 Factor" asset basket with factor assets individually constrained.* $\kappa = 1.0$.

largely similar to that of the previous results in the 2 basic asset case(i.e., Figure 4.2.4(b)), and the same is true for the other NN strategies with the "2 Factor" asset basket and $limit\_type = indiv$. The NN strategies with more lenient factor asset constraints showed controls with a much larger region in which the minimum withdrawal is made. This is likely due to the higher average wealth levels paired with riskier asset allocations in these strategies. The general "bang-bang" nature of the withdrawal control with a downward-sloping border is common to all tested NN strategies.

The wealth percentile paths, plotted for the "2 Factor" asset basket with $limit\_type = indiv$ in 5.5.3(b), show somewhat different behavior than in the 2 basic asset case (Figure 4.2.5(b)). All paths show higher levels of wealth at the 5th, 50th, and 95th percentiles. We attribute this to the higher expected returns that can be enjoyed by the investor when investing in factor assets. Each path also turns downward more sharply than in the 2 basic asset case. We attribute this to the inclusion of the T30 asset, which is essentially a cash equivalent. When the investor reaches near the end of the investment period with

sufficient levels of wealth, the NN strategy moves allocation towards the safest asset, T30, while continuing to withdraw wealth. At high enough levels of wealth, the risk of the investor reaching a shortfall at $t = T$ becomes negligible even when withdrawing the maximum amount. This phenomenon can be observed in all NN strategies that include the T30 asset.

The asset allocations are represented in heat maps 5.5.2 (a)-(e), as well as the median allocation plot in 5.5.3(a). As expected, the factor assets reach the maximum allocation allowed by the constraint function in large regions of the time-wealth space. The "Mkt" asset is relegated to an asset of last resort when the NN attempts to recover the portfolio at low levels of wealth. Similar to the control for the basic 2 asset strategy described in Chapter 4, the total allocation in equity-based assets never exceeds 0.40. However, in contrast to the percentile plots for the basic asset strategy in Figure 4.2.5(a), Figure 5.5.3(a) shows that the median factor-based strategy moves wealth into the risk-free assets much earlier in the investment period. In summary, it can be said that the dynamic factor strategies tend to invest significantly in the relatively risky factor assets for early phases of the investment period to allow the portfolio to appreciate, and then lock in gains by moving to the least risky assets.



(a) *Proportions in each candidate asset at median wealth*

(b) *Percentiles wealth*

(c) *Percentiles withdrawals*

FIGURE 5.5.3: *Percentile plots for the NN Strategy with "2 Factor" asset basket (Size, Value, and basic assets) and limit_type = indiv. $\kappa = 1.0$.*

Returning to the results displayed in Table 5.5.1, we see that the performance of the NN strategies in DS1 appears to be very robust on the out-of-sample testing data set. All factor NN strategies have significantly better expected shortfall results than the basic asset strategies while maintaining average withdrawals that are on par. Both performance measures are significantly higher for all NN strategies than for the constant benchmarks. With a test data set that is from the same distribution, this is perhaps expected.

In summary, the results presented here show that the additional inclusion of factor assets and T30 indeed provides the investor the opportunity to improve portfolio performance by providing exposure to assets with higher expected returns, as well as new ways to manage risk, such as allocating in T30 near the end of the investment horizon. At the $\kappa = 1$ risk-reward weighting, the factor-based NN strategies appear to provide similar average withdrawals and significantly improved expected shortfall compared to the basic 2 asset strategy presented in the previous chapter. This provides an attractive risk and reward trade-off. In the next section, we investigate if this result holds for an out-of-distribution test.

5.5.3 Results: DS2

In this section, we will focus on reviewing the results of the out-of-distribution test in DS2. The DS2 data set uses the the date range of 1963:07-1991:12 to generate training data and the date range of 1992:01-2022:12 for testing data. Recall that the "4 Factor" asset basket is omitted from these experiments due to the lack of data availability for the the Low Volatility factor.

As in the DS1 experiments, all factor-based NN strategies outperformed the basic asset NN strategies in both training and testing performance. The NN controls trained on DS2 are qualitatively very similar to those trained on DS1. The "3 Factor" (Momentum, Size, Value, and basic assets) NN strategies show the highest overall performance. Upon inspecting the controls, we can see that the "3 Factor" strategies all allocate heavily into the Momentum factor. The Momentum factor has a higher expected return than any other asset with standard deviation only slightly higher than that of the "Mkt" asset. The cumulative returns of the Momentum factor can be seen to outpace the other assets in Figure 5.4.2, and the strategy to heavily invest in Momentum is intuitive.

Regarding the relative performance of NN strategies with different factor asset constraints, neither type of asset constraint seemed to improve test performance for factor strategies. Further experimentation would be needed in order to make any conclusions on the efficacy of the addition factor asset constraints in improving the NN strategies' robustness.

Once again, we select the "2 Factor," $limit\_type = indiv.$ strategy as an example to focus on in analyzing the factor strategies' performance relative to the basic asset strategies in Figure 5.5.4. We observe that the factor strategy produces a EW-ES frontier to the right of that produced by the basic asset strategy, even when tested on an out-of-distribution

test data set drawn from a different data range. The constant factor strategy actually underperformed the basic asset constant benchmarks in this case.

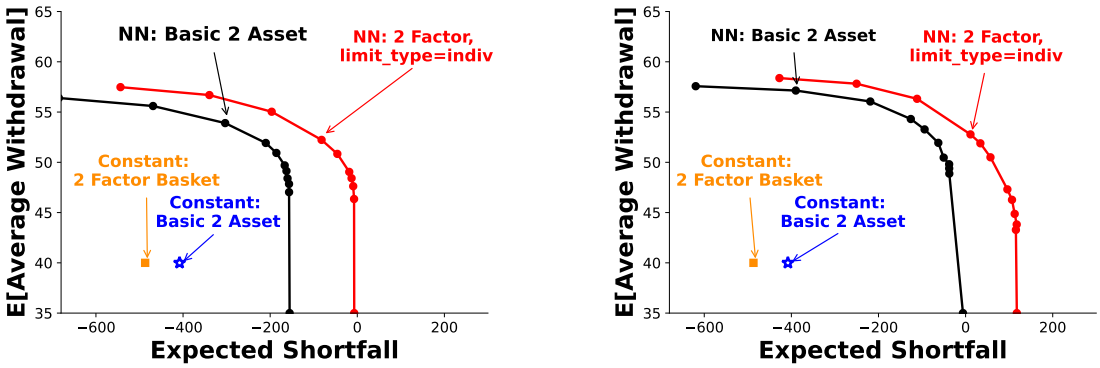
Results: DS2

NN Strategy		Training			Testing		
Asset Basket	*limit_type*	ES (5%)	Avg. q_i	Obj. Fun.	ES (5%)	Avg. q_i	Obj. Fun.
2 Basic	none	-203.3	51.7	1400.6	-117.2	54.1	1560.1
3 Basic		-166.2	51.3	1424.3	-88.2	53.1	1546.4
2 Factor	none	-14.8	52.3	1607.3	129.6	52.6	1761.7
	group	-65.0	52.8	1572.1	41.6	53.2	1692.2
	indiv.	-60.5	52.2	1558.3	52.3	53.0	1695.1
3 Factor	none	482.3	52.5	2111.0	683.3	51.1	2266.7
	group	235.6	52.7	1869.0	400.4	51.1	1985.5
	indiv.	108.7	51.9	1717.2	235.5	51.0	1817.0

Constant Benchmark	Training			Testing		
Asset Basket	ES (5%)	Avg. q_i	Obj. Fun.	ES (5%)	Avg. q_i	Obj. Fun.
2 Basic	-406.40	40	833.6	-242.5	40	997.5
3 Basic	-424.81	40	815.2	-304.2	40	935.8
2 Factor	-441.59	40	798.4	-195.5	40	1044.5
3 Factor	-405.45	40	834.6	-160.5	40	1079.5

TABLE 5.5.2: *Computational results for DS2, defined in Table 5.4.2. NN methodology defined in Section 5.2. Investment scenario presented in Table 5.3.2. Constant benchmark strategies described in 5.5.1.* $\kappa = 1.0$.

In summary, the experimental results from DS2 show that the NN factor strategies continue to provide improved investment performance when compared to the basic asset NN strategies.



(a) *DS2, Training*

(b) *DS2, Testing*

FIGURE 5.5.4: *EW-ES frontiers of controls generated by NN model for the basic asset basket and the 2-factor asset basket with limit_type = indiv.. Computational results for DS2, defined in Table 5.4.2. NN methodology defined in Section 5.2. Investment scenario presented in Table 5.3.2. Points showing the performance of associated constant strategy benchmarks are also included. The constant strategy with "Basic 2 Asset" portfolio is essentially the Bengen 4% strategy with fixed 4% withdrawals and constant allocation of 30% in stocks, which was found to be the best performing constant allocation. The constant benchmark strategy for the 2-factor basket is described in subsection 5.5.1. Note that pareto optimality is not guaranteed for points calculated using the test data set, which is why there may be subtle irregularities in the EW-ES frontiers plotted in Subfigure (b).*

Chapter 6

Conclusion

6.1 Conclusion

In this thesis, we proposed a novel neural network (NN) architecture to efficiently and accurately compute the optimal decumulation strategy for retirees with DC pension plans. After developing the NN framework, we demonstrated its accuracy by comparing it to the solution to the same problem, as computed by a provably convergent HJB PDE method. We then extended the NN framework to find the optimal dynamic factor investing strategy for the decumulation problem, whereby we find promising evidence of the usefulness of factor investing for DC plan investors managing the decumulation problem.

The increasing prevalence of DC pension plans over traditional DB pension plans make the decumulation problem ever more critical for DC plan investors. There is extensive literature on devising strategies for this problem. In particular, we discuss a Hamilton-Jacobi-Bellman (HJB) Partial Differential Equation (PDE) based approach that can be shown to converge to an optimal solution for a dynamic withdrawal/allocation strategy. This provides an attractive balance of risk management and withdrawal efficiency for retirees. In this paper, we seek to build upon this approach by developing a new, more versatile framework using NNs to solve the decumulation problem.

We conduct computational experiments to demonstrate the accuracy and robustness of the proposed NN solution, utilizing the unique opportunity to compare NN solutions with the HJB results as a ground truth. Of particular noteworthiness is that the continuous function approximation from the NN framework is able to approximate a bang-bang control with high accuracy. We extend our experiments to establish the robustness of our approach, testing the NN control's performance on both synthetic and historical data sets.

We demonstrate that the proposed NN framework produced solution accurately approximates the ground truth solution. We also note the following advantages of the proposed NN framework:

(i) The NN method is data driven, and does not require postulating and calibrating a parametric model for market processes.

(ii) The NN method directly estimates the low dimensional control by solving a single unconstrained optimization problem, avoiding the problems associated with dynamic programming methods, which require estimating high dimensional conditional expectations (see Van Staden et al. (2023)).

(iii) The NN formulation maintains its simple structure (discussed in Section 3.4.2), immediately extendable to problems with more frequent rebalancing and/or withdrawal events. In fact, the problem presented in (3.1.22) requires each control NN to have only two hidden layers for 30 rebalancing and withdrawal periods.

(iv) The approximated control maintains continuity in time and/or space, provided it exists, or otherwise provides a smooth approximation. Continuity of the allocation control p is an important practical consideration for any investment policy.

Due to the ill-posedness of the stochastic optimal control problem in the region of high wealth near the end of the decumulation horizon, we observe that the NN allocation can appear to be very different from the HJB PDE solution. We note, however, that both strategies yield indistinguishable performance when assessed with the expected withdrawal and ES reward-risk criteria. In other words, these differences hardly affect the objective function value, a weighted reward and risk value. In the region of high wealth near the end of time horizon, the retiree is free to choose whether to invest 100% in stocks or 100% in bonds, since this has negligible effect on the objective function value (or reward-risk consideration).[1]

In Chapter 5 of this thesis, we extended the proposed NN framework to include an arbitrary number of additional candidate assets. This expanded problem space is not computationally feasible with a HJB PDE method. We also defined new activation functions to encode additional constraints on the allocations of these assets. The extended framework

[1]This can be termed the *Warren Buffet* effect. Buffet is the fifth richest human being in the world. He is 92 years old. Buffet can choose any allocation strategy, and will never run out of cash.

was applied to the dynamic factor investing problem with decumulation in order to concretely demonstrate the versatility of the NN approach, as well as investigate the usefulness of factor investing for investors facing the decumulation problem.

We performed experiments on two different pairs of training/testing data sets. The results of these experiments yielded promising results for all factor portfolios tested on an out-of-sample data set. Factor portfolios also performed well on an out-of-distribution test, yielding more efficient withdrawals and better risk management than the NN strategies with only basic assets and the constant benchmark. Portfolios including the Momentum factor showed especially improved performance, which we ascribe to the outsize returns of the Momentum factor over the chosen data period.

While our factor investing results are promising, this cannot be taken as conclusive evidence for dynamic factor investing's reliability. Even though the long-term nature of our problem may help overcome the cyclicality and regime-dependency of factor performance (Bender et al., 2013), it has been shown that even portfolios constructed with multiple factors of low correlation may not reduce macroeconomic risks (Amenc et al., 2019). Furthermore, while our out-of-distribution robustness test yielded promising results, the *publication effect*, which refers to the phenomenon of factor premia decreasing or disappearing after publication (Dimson et al., 2017), demands that any prudent investor be suspicious of factor models learned from historical data. More research would be needed to ascertain whether the long-term investment horizon of our problem truly allows our framework to sufficiently overcome the cyclicality of factor assets' performance to make dynamic factor investing a recommendable strategy for retirees.

One aim of the factor investing investigation was to impose additional constraints on factor asset allocations as a means to increase asset diversification. The resulting optimal investing strategies were indeed more diversified, but no strong conclusions could be made on whether these constraints made the strategies more robust. This could in part be due to the arbitrary constraints chosen. More experimentation would be needed, perhaps also with different factors, in order to determine how such an NN framework could be induced to create a more diversified and robust investment strategy.

In summary, the investigation of dynamic factor investing using our proposed NN framework effectively demonstrated the versatility of the novel NN approach developed in this thesis to solve the optimal decumulation problem. We expect that the NN approach can be adapted to other formulations of the retirement and investing problem and that it can handle problems of even higher complexity. We leave the further extension of this methodology to future work.

References

N. Amenc, M. Esakia, F. Goltz, and B. Luyten. Macroeconomic risks in equity factor investing. *The Journal of Portfolio Management*, 45(6):39–60, 2019.

A. Anarkulova, S. Cederburg, and M. S. O'Doherty. Stocks for the long run? evidence from a broad sample of developed markets. *Journal of Financial Economics*, 143(1): 409–433, 2022.

A. Ang. Factor capacity by the numbers, Dec 2019. URL https://www.blackrock.com/us/individual/investment-ideas/what-is-factor-investing/factor-commentary/andrews-angle/factor-capacity-by-the-numbers.

C. S. Asness, T. J. Moskowitz, and L. H. Pedersen. Value and momentum everywhere. *The Journal of Finance*, 68(3):929–985, 2013. ISSN 00221082, 15406261. URL http://www.jstor.org/stable/42002613.

J. Bender, R. Briand, D. Melas, and R. A. Subramanian. Foundations of factor investing. *Available at SSRN 2543990*, 2013.

J. Bender, X. Sun, R. Thomas, and V. Zdorovtsov. The promises and pitfalls of factor timing. *The Journal of Portfolio Management*, 44(4):79–92, 2018.

W. Bengen. Determining withdrawal rates using historical data. *Journal of Financial Planning*, 7:171–180, 1994.

T. Bernhardt and C. Donnelly. Pension decumulation strategies: A state of the art report. Technical Report, Risk Insight Lab, Heriot Watt University, 2018.

D. Bertsekas and S. E. Shreve. *Stochastic optimal control: the discrete-time case*, volume 5. Athena Scientific, 1996.

D. Bertsekas and J. N. Tsitsiklis. *Neuro-dynamic programming*. Athena Scientific, 1996.

T. Bjork and A. Murgoci. A general theory of Markovian time inconsistent stochastic control problems. SSRN 1694759, 2010.

T. Bjork and A. Murgoci. A theory of Markovian time inconsistent stochastic control in discrete time. *Finance and Stochastics*, 18:545–592, 2014.

D. Blitz, G. Baltussen, and P. van Vliet. When equity factors drop their shorts. *Financial Analysts Journal*, 76(4):73–99, 2020. doi:10.1080/0015198X.2020.1779560. URL https://doi.org/10.1080/0015198X.2020.1779560.

E. B. Boukherouaa, K. AlAjmi, J. Deodoro, A. Farias, and R. Ravikumar. Powering the digital economy: Opportunities and risks of artificial intelligence in finance. *IMF Departmental Papers*, 2021(024):A001, 2021. doi:10.5089/9781589063952.087.A001. URL https://www.elibrary.imf.org/view/journals/087/2021/024/article-A001-en.xml.

H. Buehler, L. Gonon, J. Teichmann, and B. Wood. Deep hedging. *Quantitative Finance*, 19(8):1271–1291, 2019.

M. M. Carhart. On persistence in mutual fund performance. *The Journal of finance*, 52 (1):57–82, 1997.

Y. Chen and J. W. L. Wan. Deep neural network framework based on backward stochastic differential equations for pricing and hedging american options in high dimensions. *Quantitative Finance*, 21(1):45–67, 2021. doi:10.1080/14697688.2020.1788219. URL https://doi.org/10.1080/14697688.2020.1788219.

P. Cogneau and V. Zakamouline. Block bootstrap methods and the choice of stocks for the long run. *Quantitative Finance*, 13(9):1443–1457, 2013.

R. Cont and C. Mancini. Nonparametric tests for pathwise properties of semimartingales. *Bernoulli*, 17:781–813, 2011.

G. Coqueret and T. Guida. *Machine Learning for Factor Investing: Python Version*. CRC Press, 2023.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

D.-M. Dang and P. A. Forsyth. Continuous time mean-variance optimal portfolio allocation under jump diffusion: a numerical impulse control approach. *Numerical Methods for Partial Differential Equations*, 30:664–698, 2014.

D.-M. Dang and P. A. Forsyth. Better than pre-commitment mean-variance portfolio allocation strategies: a semi-self-financing Hamilton-Jacobi-Bellman equation approach. *European Journal of Operational Research*, 250:827–841, 2016.

M. H. A. Davis and M. Farid. *Piecewise-Deterministic Processes and Viscosity Solutions*, pages 249–268. Birkhäuser Boston, Boston, MA, 1999. ISBN 978-1-4612-1784-8. doi:10.1007/978-1-4612-1784-8_15. URL https://doi.org/10.1007/978-1-4612-1784-8_15.

F. Della Santa and S. Pieraccini. Discontinuous neural networks and discontinuity learning. *Journal of Computational and Applied Mathematics*, 419:114678, 2023. ISSN 0377-0427. doi:https://doi.org/10.1016/j.cam.2022.114678. URL https://www.sciencedirect.com/science/article/pii/S0377042722003430.

H. Dichtl, W. Drobetz, and M. Wambach. Testing rebalancing strategies for stock-bond portfolios across different asset allocations. *Applied Economics*, 48(9):772–788, 2016.

H. Dichtl, W. Drobetz, H. Lohre, C. Rother, and P. Vosskamp. Optimal timing and tilting of equity factors. *Financial Analysts Journal*, 75(4):84–102, 2019. doi:10.1080/0015198X.2019.1645478. URL https://doi.org/10.1080/0015198X.2019.1645478.

E. Dimson, P. Marsh, and M. Staunton. Factor-based investing: The long-term evidence. *The Journal of Portfolio Management*, 43(5):15–37, 2017.

T. Dutt and M. Humphery-Jenner. Stock return volatility, operating performance and stock returns: International evidence on drivers of the 'low volatility' anomaly. *Journal of Banking Finance*, 37(3):999–1017, 2013. ISSN 0378-4266. doi:https://doi.org/10.1016/j.jbankfin.2012.11.001. URL https://www.sciencedirect.com/science/article/pii/S0378426612003408.

W. Edelberg and N. Steinmetz-Silber. High mortgage rates are probably here for a while, Apr 2023. URL https://www.brookings.edu/articles/high-mortgage-rates-are-probably-here-for-a-while/. https://www.brookings.edu/articles/high-mortgage-rates-are-probably-here-for-a-while/.

E. F. Fama and K. R. French. Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics*, 33(1):3–56, 1993. ISSN 0304-405X. doi:https://doi.org/10.1016/0304-405X(93)90023-5. URL https://www.sciencedirect.com/science/article/pii/0304405X93900235.

Federal Reserve Bank of Atlanta. Survey of business uncertainty, 2023. https://www.atlantafed.org/research/surveys/business-uncertainty.

G. Feng, S. Giglio, and D. Xiu. Taming the factor zoo: A test of new factors. *The Journal of Finance*, 75(3):1327–1370, 2020.

G. Feng, J. He, N. G. Polson, and J. Xu. Deep learning in characteristics-sorted factor models, 2022.

W. Fleming and R. Rishel. *Deterministic and stochastic optimal control*. Springer New York, 1975.

P. A. Forsyth. Multi-period mean CVAR asset allocation: Is it advantageous to be time consistent? *SIAM Journal on Financial Mathematics*, 11:2:358–384, 2020.

P. A. Forsyth. A stochastic control approach to defined contribution plan decumulation: "the nastiest, hardest problem in finance". *North American Actuarial Journal*, 26(2): 227–251, 2022.

P. A. Forsyth and G. Labahn. $\epsilon-$Monotone Fourier methods for optimal stochastic control in finance. *Journal of Computational Finance*, 22:4:25–71, 2019.

P. A. Forsyth and K. R. Vetzal. Optimal asset allocation for retirement savings: deterministic vs. time consistent adaptive strategies. *Applied Mathematical Finance*, 26:1:1–37, 2019.

P. A. Forsyth, K. R. Vetzal, and G. Westmacott. Optimal performance of a tontine overlay subject to withdrawal constraints. *arXiv 2211.10509*, 2022.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

V. Haddad, S. Kozak, and S. Santosh. Factor Timing. *The Review of Financial Studies*, 33(5):1980–2018, 02 2020. ISSN 0893-9454. doi:10.1093/rfs/hhaa017. URL https://doi.org/10.1093/rfs/hhaa017.

M. Haghbin and K. Masse. Invesco 7th global factor investing study. *Invesco*, Mar 2022.

J. Han and W. E. Deep learning approximation for stochastic control problems. *CoRR*, abs/1611.07422, 2016. URL http://arxiv.org/abs/1611.07422.

C. R. Harvey, Y. Liu, and H. Zhu. ... and the Cross-Section of Expected Returns. *The Review of Financial Studies*, 29(1):5–68, 10 2015. ISSN 0893-9454. doi:10.1093/rfs/hhv059. URL https://doi.org/10.1093/rfs/hhv059.

S. Homer and R. Sylla. *A History of Interest Rates*. Wiley, New York, 2005.

K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

C. Huré, H. Pham, A. Bachouch, and N. Langrené. Deep neural networks algorithms for stochastic control problems on finite horizon: Convergence analysis. *SIAM Journal on Numerical Analysis*, 59(1):525–557, 2021. doi:10.1137/20m1316640. URL https://doi.org/10.1137%2F20m1316640.

V. E. Ismailov. A three layer neural network can represent any multivariate function. *Journal of Mathematical Analysis and Applications*, 523(1):127096, 2023.

R. Israel and T. J. Moskowitz. The role of shorting, firm size, and time on market anomalies. *Journal of Financial Economics*, 108(2):275–301, 2013. doi:10.1016/j.jfineco.2012.11. URL https://ideas.repec.org/a/eee/jfinec/v108y2013i2p275-301.html.

I. Karatzas and S. Shreve. *Brownian motion and stochastic calculus*, volume 113. Springer Science & Business Media, 1991.

D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv*, page arXiv:1412.6980, 2014.

D. E. Kirk. *Optimal control theory: An introduction*. Dover Publications, 2016.

S. G. Kou. A jump-diffusion model for option pricing. *Management Science*, 48:1086–1101, 2002.

S. G. Kou and H. Wang. Option pricing under a double exponential jump diffusion model. *Management Science*, 50:1178–1192, 2004.

M. Laurière, O. Pironneau, et al. Performance of a markovian neural network versus dynamic programming on a fishing control problem. *arXiv preprint arXiv:2109.06856*, 2021.

Y. Li and P. A. Forsyth. A data driven neural network approach to optimal asset allocation for target based defined contribution pension plans. *Insurance: Mathematics and Economics*, 86:189–204, 2019.

Y. Lin, R. MacMinn, and R. Tian. De-risking defined benefit plans. *Insurance: Mathematics and Economics*, 63:52–65, 2015.

B.-J. MacDonald, B. Jones, R. J. Morrison, R. L. Brown, and M. Hardy. Research and reality: A literature review on drawing down retirement financial savings. *North American Actuarial Journal*, 17:181–215, 2013.

R. MacMinn, P. Brockett, J. Wang, Y. Lin, and R. Tian. The securitization of longevity risk and its implications for retirement security. In O. S. Mitchell, R. Maurer, and P. B. Hammond, editors, *Recreating Sustainable Retirement*, pages 134–160. Oxford University Press, Oxford, 2014.

C. Mancini. Non-parametric threshold estimation models with stochastic diffusion coefficient and jumps. *Scandinavian Journal of Statistics*, 36:270–296, 2009.

R. Marler and J. Arora. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26:369–395, 04 2004. doi:10.1007/s00158-003-0368-6.

C. Ni, Y. Li, P. A. Forsyth, and R. Carroll. Optimal asset allocation for outperforming a stochastic benchmark target. *Quantitative Finance*, 22(9):1595–1626, 2022a. doi:10.1080/14697688.2022.2072233. URL https://doi.org/10.1080/14697688.2022.2072233.

C. Ni, Y. Li, P. A. Forsyth, and R. Carroll. Optimal asset allocation for outperforming a stochastic benchmark target. *Quantitative Finance*, 22(9):1595–1626, 2022b.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

A. Patton, D. Politis, and H. White. Correction to: automatic block-length selection for the dependent bootstrap. *Econometric Reviews*, 28:372–375, 2009.

W. D. Pfau. An overview of retirement income planning. *Journal of Financial Counseling and Planning*, 29:1:114:120, 2018.

S. Pfeiffer, J. R. Salter, and H. E. Evensky. Increasing the sustainable withdrawal rate using the standby reverse mortgage. *Journal of Financial Planning*, 26:12:55–62, 2013.

D. Politis and J. Romano. The stationary bootstrap. *Journal of the American Statistical Association*, 89:1303–1313, 1994.

D. Politis and H. White. Automatic block-length selection for the dependent bootstrap. *Econometric Reviews*, 23:53–70, 2004.

W. B. Powell. *From Reinforcement Learning to Optimal Control: A Unified Framework for Sequential Decisions*, pages 29–74. Springer International Publishing, Cham, 2021. ISBN 978-3-030-60990-0. doi:10.1007/978-3-030-60990-0_3. URL https://doi.org/10.1007/978-3-030-60990-0_3.

B. Ritholz. Tackling the 'nastiest, hardest problem in finance'. www.bloomberg.com/view/articles/2017-06-05/tackling-the-nastiest-hardest-problem-in-finance, 2017.

R. T. Rockafellar and S. Uryasev. Optimization of conditional value-at-risk. *Journal of Risk*, 2:21–42, 2000.

S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

J. Scott, W. Sharpe, and J. Watson. The 4% rule – at what price? *Journal of Investment Management*, 7(3):31–48, 2009.

L. Scott and S. Cavaglia. A wealth management perspective on factor premia and the value of downside protection. *Journal of Portfolio Management*, 43(3):33–41, 2017.

S. P. Sethi. *Optimal Control Theory*. Number 978-3-319-98237-3 in Springer Books. Springer, August 2019. ISBN ARRAY(0x4c1490c0). doi:10.1007/978-3-319-98237-3. URL https://ideas.repec.org/b/spr/sprbok/978-3-319-98237-3.html.

W. F. Sharpe. Capital asset prices: A theory of market equilibrium under conditions of risk*. *The Journal of Finance*, 19(3):425–442, 1964. doi:https://doi.org/10.1111/j.1540-6261.1964.tb02865.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1964.tb02865.x.

H. M. Shefrin and R. H. Thaler. The behavioral life-cycle hypothesis. *Economic Inquiry*, 26(4):609–643, 1988. doi:https://doi.org/10.1111/j.1465-7295.1988.tb01520.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1465-7295.1988.tb01520.x.

J. Simonian and A. Martirosyan. Sharpe parity redux. *The Journal of Portfolio Management*, 48(4):183–193, 2022.

M. Strub, D. Li, and X. Cui. An enhanced mean-variance framework for robo-advising applications. SSRN 3302111, 2019.

R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3:9–44, 1988.

C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu. A survey on deep transfer learning. *arXiv 1808.01974*, 2018.

P. Tankov and R. Cont. *Financial Modelling with Jump Processes*. Chapman and Hall/CRC, New York, 2009.

K. H. Tsang and H. Y. Wong. Deep-learning solution to portfolio selection with serially-dependent returns. *SSRN*, 10.2139, 2020. URL https://ssrn.com/abstract=3396246.

U.S. Bureau of Labor Statistics. Employee benefits survey: Latest numbers, 2022. https://www.bls.gov/ebs/latest-numbers.htm.

P. Van Staden, Y. Li, and P. A. Forsyth. A data-driven neural network approach to dynamic factor investing with transaction costs, 01 2022.

P. Van Staden, P. A. Forsyth, and Y. Li. Beating a benchmark: dynamic programming may not be the right numerical approach. *SIAM Journal on Financial Mathematics*, 14:2:407–451, 2023.

E. Vigna. On efficiency of mean-variance based portfolio selection in defined contribution pension schemes. *Quantitative Finance*, 14:237–258, 2014.

E. Vigna. Tail optimality and preferences consistency for intertemporal optimization problems. Working paper no. 502 , Collegio Carlo Alberto, Università Degli Studi di Torino, 2017.

R. Williams and C. Kawashima. Beyond the 4% rule: How much can you spend in retirement? *Charles Schwab Brokerage*, 2023. URL https://www.schwab.com/learn/story/beyond-4-rule-how-much-can-you-spend-retirement. https://www.schwab.com/learn/story/beyond-4-rule-how-much-can-you-spend-retirement.